NATIONAL TECHNICAL UNIVERSITY OF ATHENS
School of Civil Engineering
Laboratory for Earthquake Engineering

Application of Machine Learning Algorithms to Rocking Problems

Zeinep Achmet

A dissertation submitted in partial fulfillment
of the requirements for the degree of

MSc in Analysis & Design of Earthquake Resistant Structures

Supervisor
Dr. Michalis Fragiadakis
Assistant Professor NTUA

Athens, June, 2019

Research topics : Earthquake Engineering, Machine Learning

Key words : rocking structures, rigid blocks, seismic response, machine learning, support vector machines, Gaussian process regression, k-nearest neighbours, random forest

Zeinep Achmet, Dipl (School of Civil Engineering,NTUA,2017)

Dr. Michalis Fragiadakis, Assistant Professor NTUA

Application of Machine Learning Algorithms to Rocking Problems

MSc in Analysis & Design of Earthquake Resistant Structures

Master's Dissertation

Laboratory for Earthquake Engineering, School of Civil Engineering, National Technical University of Athens, Greece, 2019.

ABSTRACT

This master's dissertation examines the application of machine learning learning algorithms to rocking problems. Initially, classification algorithms such as, support vector machines, k-nearest neighbours and random forests are trained to predict accurately the overturning condition of rigid blocks subjected to one-sine base excitation. For the same case, regression models such as Gaussian process regression, support vector regression and random forest are used to predict the overturning ratio of the blocks. The performance of the algorithms are compared and the best performing algorithms are applied to the case of rigid blocks subjected to near-fault ground motion. The training data-set for both cases are prepared solving the equation of motion of rigid blocks with MATLAB's ODE23s solver.

ACKNOWLEDGEMENTS

I would first like to thank my thesis supervisor Dr. Michalis Fragiadakis, Assistant Professor of the Laboratory for Earthquake Engineering of NTUA. The door to Dr. Fragiadakis office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this dissertation to be my own work, but steered me in the right direction whenever he thought I needed it.

I would also like to thank Spyros Diamantopoulos, PhD student at the Laboratory for Earthquake Engineering of NTUA. Without his passionate participation and input, this master's dissertation could not have been successfully conducted.

Finally, I must express my very profound gratitude to my parents and to my sister for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF TABLES

Listings

Chapter 1: Introduction

Machine learning is undeniably one of the most influential and powerful technologies in today's world. There is no doubt, it will continue to be making headlines for the foreseeable future, as a tool for turning information into knowledge. In the past 50 years, there has been an explosion in information generation related to all aspects of life including all engineering disciplines. This mass of data is useless unless we analyze it and find the patterns hidden within. Machine learning techniques are used to automatically find the valuable underlying patterns within complex data that we would otherwise struggle to discover. The hidden patterns and knowledge about a problem can be used to predict future events and perform all kinds of complex decision making.

In recent years, machine learning has found a wide range of applications in different fields of civil engineering problems such as structural engineering, construction management, hydrology, hydraulic engineering, geotechnical engineering, environmental engineering, transportation engineering, coastal and ocean engineering and materials of construction. The increase in machine learning studies with great acceleration shows that the use of machine learning in civil engineering will increase in the coming years.

The purpose of this master's dissertation is to investigate the application of machine learning algorithms to rocking problems. The rocking response of bodies

subjected to earthquake ground motion is investigated in many researches the last years. Rocking motion, established in either the superstructure in the form of a 2-point stepping mechanism (structural rocking) or resulting from rotational motion of the foundation on the soil (foundation rocking), is considered an effective, low-cost base isolation technique. The seismic behaviour of a wide variety of structures can be characterized by the rocking response of rigid blocks.

Specifically in this dissertation, a number of machine learning models are trained to predict the seismic response of rocking blocks subjected to pulse-like excitation. In order to prepare the data for training, the equation of motion of various rigid blocks is solved using standard ODE solvers available in MATLAB for varying geometric properties and seismic excitation characteristics. Then, machine learning algorithms are fed with this training data to make accurate predictions about the rocking response of new blocks and their performances are compared. The study is extended to blocks subjected to near-fault ground motion records using the best performing algorithms.

In chapter 2, the theory behind the support vector machine, k-nearest neighbours, random forest and Gaussian process regression algorithms which are used for classification and regression problems is discussed. The performance evaluation and the estimation of the generalization error, the k-fold cross-validation method, the optimization of the algorithm hyper-parameters, as well as the performance evaluation metrics are also discussed in this chapter.

In chapter 3, the types of seismic response of a free-standing block under a seismic excitation are presented. Specifically, sliding and uplifting and the rocking

motion of a rigid block, the equation of the rocking motion and the two overturning modes under one-sine pulse are discussed.

In chapter 4, the application of machine learning algorithms to rocking problems is presented. Specifically, the preparation of the training data consisting of rigid blocks subjected to one-sine pulse and the applied classification and regression models are described. The implementation of each model and their performance comparison are also discussed.

In chapter 5, the application of machine learning algorithms to rocking blocks under near-fault ground motion records is investigated. The preparation of the training data is described. The best performing algorithm for regression is applied for this case. Also, predictor importance estimation is accomplished.

In chapter 6, the conclusions of this research are discussed and further research ideas are proposed.

Chapter 2: Machine Learning Algorithms

2.1 Introduction

Machine learning is the scientific study of algorithms and statistical models that computer systems use to effectively perform a specific task without using explicit instructions, relying on models and inference instead. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model of sample data, known as training data, in order to make predictions or decisions without being explicitly programmed to perform the task [1].

Machine learning algorithms are often categorized as *supervised* or *unsupervised*. In supervised learning, each training example is a pair consisting of an input feature and a desired output value and the algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples [2]. In unsupervised learning, the training data has only the input values and the algorithm identifies commonalities in the data and reacts based on the presence or absence of such commonalities in each new piece of data. Estimation of the importance of the input features for the prediction of the output, namely feature selection is another task which can be accomplished with learning algorithms.

In this dissertation, the supervised learning method is used for both *classification* and *regression*. In classification problems, the outputs are restricted to a

limited set of values (a category or a group). Regression problems are named for their continuous outputs, meaning they may have any value within a range.

There are plenty of algorithms for either task. In this dissertation, support vector machine, k-nearest neighbor, random forest and Gaussian process regression algorithms are used for classification and regression problems. Furthermore, the random forest algorithm is used for input feature importance estimation. The theory behind these algorithms as well as their application to rocking problems will be discussed in the next sections.

2.2 Machine Learning Algorithms

2.2.1 Support Vector Machines

2.2.1.1 Support Vector Machines for Classification

A support vector machine (SVM) is a discriminative classifier formally defined by a separating hyper-plane (i.e., the decision boundary). In other words, given labeled training data, the algorithm outputs an optimal hyper-plane which categorizes new examples. In two dimensional space this hyper-plane is a line dividing a plane in two parts where each class lay in either side.

The basic idea of a support vector machine is to find the optimal hyper-plane (maximum margin) for linearly separable patterns, extend to linearly inseparable patterns by transforming the original data to map into new space with the kernel function and use support vector machine algorithm for pattern recognition.

Input: set of training pair samples, input features (i.e, predictors) $x_1, x_2...x_n$ and the output result (i.e, response) y_i. Typically, there can be lots of input features x_i.

Output: set of weights w_i, one for each feature, whose linear combination predicts the value of yi.

The separation between the hyper-plane and the closest data point for a given weight vector w and bias b is called margin of separation. Optimal hyper-plane is the particular hyper-plane for which the margin of separation is maximized.

The optimization of maximizing the margin is a quadratic programming problem and it is used to reduce the number of weights that are nonzero to just few that correspond to the important features which are the support vectors (they support the separation hyper-plane). Support vectors are the data points that lie closest to the separation hyper-plane and are the most difficult to classify. They have direct bearing on the optimum location of the hyper-plane. The mathematical formulation of the support vector machines according to [3] and [4] is presented in the next subsections.

2.2.1.2 Binary Classification-Linearly Separable Data

The training data is a set of points (vectors) x_i along with their categories y_i, for some dimension d, the $x_i \in \Re^d$ and the y_i=$\pm$1. The equation of the hyper-plane

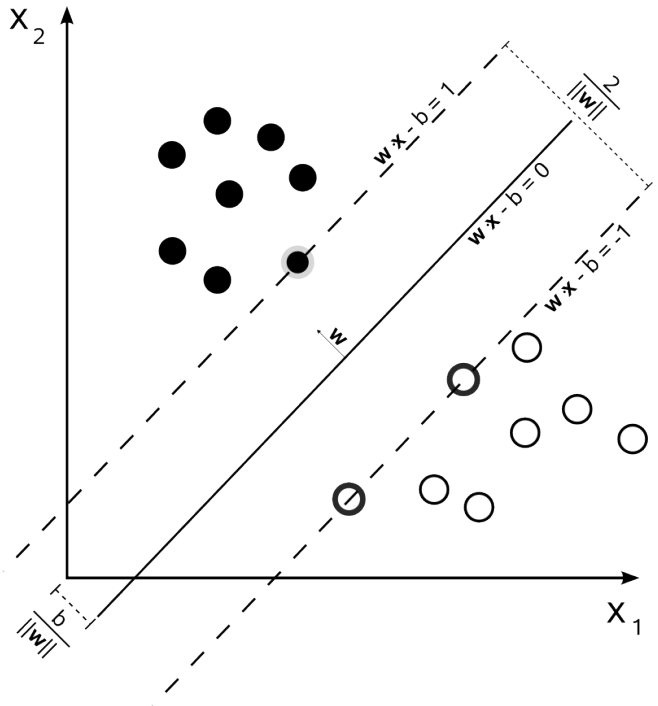


Figure 2.1: Support vectors and separation hyper-plane,two types of data points.

is:

$$f(x) = wx^T + b = 0 \tag{2.1}$$

where $w \in \Re^d$ and the bias, b is a real number.

The following problem defines the optimal separating hyper-plane. Find w and b that maximize the margin of seperation $\frac{2}{\|w\|}$, as shown in Fig.(2.1), such that for all data points (x_i, y_i),

$$y_i f(x_i) \geq 1 \tag{2.2}$$

The support vectors are the x_i on the boundary, those for which $y_i f(x_i)$=1.

For mathematical convenience, the problem is usually given as the equivalent problem of minimizing $\|w\|$. This is a quadratic programming problem which can be solved by the Lagrangian multiplier method. The Lagrangian formulation in the SVM problem is:

Primal Problem:

$$minL_p = \frac{1}{2}w^T w - \sum_{i=1}^{n} a_i y_i (x_i^T \cdot w + b) + \sum_{i=1}^{n} a_i \tag{2.3}$$

$$\text{s.t. } a_i \geq 0, \forall \text{ i},$$

where a_i are the Lagrange multipliers and n is the number of training points.

Setting the gradient of L_p to 0, we get:

$$\begin{aligned} w &= \sum_{i=1}^{n} a_i y_i x_i \\ \sum_{i=1}^{n} a_i y_i &= 0 \end{aligned} \tag{2.4}$$

Dual Problem:

$$\max L_D(a_i) = \sum_{i=1}^{n} a_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j y_i y_j (x_i^T \cdot x_j) \tag{2.5}$$

$$\text{s.t. } \textstyle\sum_{i=1}^{n} a_i y_i = 0 \text{ and } a_i \geq 0$$

It is computationally simpler to solve the Lagrangian Dual Problem. Instead of minimizing over w, b, subject to constraints involving a's, we can maximize over

a subject to the relations obtained preciously for w and b. The solution of the dual problem must satisfy the relations 2.4. By substituting for w and b in 2.3, the dependence on w and b is eliminated. We already know that the weights, w are linear combination of training inputs and outputs, x_i and y_i and the values of a. Hence, we solve for a's by differentiating the dual problem with respect to a, and setting it to zero.

$$\begin{aligned} \sum_{i=1}^{l} a_i y_i &= 0 \\ a_i &\geq 0 \end{aligned} \tag{2.6}$$

Most of the a's will turn out to have the value zero. The non-zero a's will correspond to support vectors. In this way the dimensionality of the problem is reduced.

After the calculation of the a_i, the weights w_i for the maximal margin separating hyper-plane can be determined using the equation in 2.4.

The optimal solution $(\hat{w}, \hat{b})$ enables classification of a vector z with unknown point as follows:

$$class(z) = sign(z^T \hat{w} + \hat{b}) = sign(\hat{f}(z)) \tag{2.7}$$

where $\hat{f}(z)$ is the classification score and represents the distance z from the decision boundary.

2.2.1.3 Binary Classification - Linearly Inseparable Data

In the case of non-separable data, SVM can use *soft margin*, meaning a hyper-plane that separates many, but not all data points.

There are two formulations of soft margins. Both involve adding slack variables ξ_i and a penalty parameter C.

The L^1-norm problem is:

$$\min\left(\frac{1}{2}w^T w + C\sum_{i=1}^{l}\xi_i\right) \tag{2.8}$$

such that

$$\begin{aligned} y_i f(x_i) &\geq 1 - \xi_i \\ \xi_i &\geq 0 \end{aligned} \tag{2.9}$$

The L^2-norm problem is:

$$\min\left(\frac{1}{2}w^T w + C\sum_{i=1}^{l}\xi_i^2\right) \tag{2.10}$$

subject to the same constraints.

As mentioned before, C is a penalty parameter. Increasing C places more weight on the slack variables ξ_i, meaning that the optimization attempts to make stricter separation between the classes. Equivalently, reducing C towards zero makes misclassification less important.

In an analogous manner with the case of separable data, either L^1-norm or L^2 can be solved using Lagrange multipliers. The dual problem is the same with the case of separable data (2.5), with a difference in the second constraint:

$$\begin{aligned} \sum_{i=1}^{l} a_i y_i &= 0 \\ 0 \leq a_i &\leq C \end{aligned} \tag{2.11}$$

The final set of inequalities $0 \leq a_i \leq C$, shows why C is sometimes called a box constraint. The parameter C keeps the allowable values of the Lagrange multipliers a_i in a bounded region.

2.2.1.4 Nonlinear Transformations with Kernels

In case of non linear decision boundary, the kernel method is used to gain linearly separation by mapping the data x_i, x_j to a higher dimensional space $\phi(x_i), \phi(x_j)$.

In the function we optimize (2.5), we notice the term $(x_i^T \cdot x_j)$ which shows that we do not need the exact data points, but only their inner products (measure of similarity) to compute the decision boundary. Hence, in the case we want to transform our data to a higher dimensional space there is no need to compute the exact transformation of the data $\phi(x_i)^T, \phi(x_j)$ but only the inner product of the data in the transformed space $(\phi(x_i)^T \cdot \phi(x_j))$. According to the kernel method, these inner products are defined by a kernel (similarity) function such

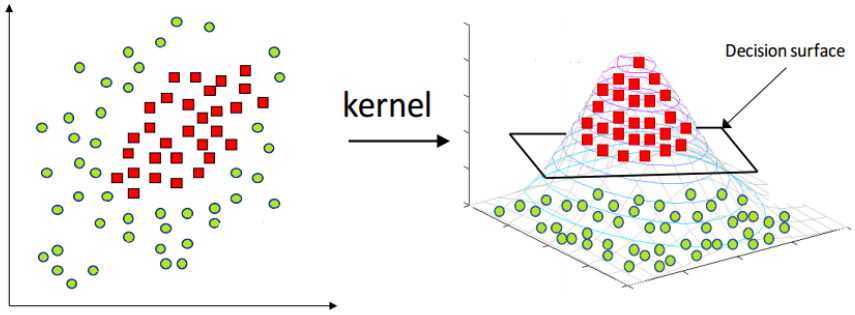


Figure 2.2: Transformation of the input data by means of a kernel function into a higher dimension feature space. a) Input feature space, b)Kernel induced high dimensional feature space.

that $K(x_i, x_j) = (\phi(x_i)^T \cdot \phi(x_j))$. The dual problem (2.5), after adding the kernel function becomes:

$$\max L_D(a_i) = \sum_{i=1}^{l} a_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} a_i a_j y_i y_j K(x_i^T \cdot x_j) \tag{2.12}$$

Some of the commonly used non linear kernel functions are:

- Gaussian (or radial basis function) :

$$K(x_1, x_2) = \exp(-\frac{\|x_1 - x_2\|}{2\sigma^2})$$

  where the width σ^2 is specified by the user. As σ^2 increases, the features x_i vary more smoothly.

- Polynomial :

$$K(x_1, x_2) = (x_1^T x_2 + 1)^p$$

where the power p is specified by the user.

- Two layer neural network :

$$\tanh \beta_0 x_1^T x_2 + \beta_1$$

which actually works only for some values of β_0 and β_1

2.2.1.5 Multiclass Classification

The multiclass classification problem can be decomposed into several binary classification tasks that can be solved efficiently using binary classifiers. One-versus-all and one-versus-one are techniques for this transformation.

One-versus-all

The simplest approach is to reduce the problem of classifying among K classes into K binary problems, where each problem discriminates a given class from the other $K-1$ classes [5]. For this approach, K binary classifiers are required, where the k^{th} classifier is trained with positive examples belonging to class k and negative examples belonging to the other $k-1$ classes. When testing an unknown example, the classifier producing the maximum ouput is considered the winner, and this class label is assigned to that example.

One-versus-one

In this approach, each class is compared to each other class [6]. A binary classifier is built to discriminate between each pair of classes, while discarding the rest

of the classes. This requires building $\frac{K(K-1)}{2}$ binary classifiers. When testing a new example, a voting is performed among the classifiers and the class with the maximum number of votes wins.

2.2.1.6 Regression

Support vector machines can also be used as a regression method, maintaining all the main features that characterize the algorithm (maximal margin). The support vector regression (SVR) uses the same principles as the SVM for classification, with only a few minor differences. First of all, because the output is a real number it becomes very difficult to predict the information at hand, which has infinite possibilities. In the case of regression, a margin of tolerance (epsilon) is set in approximation to the SVM which would have already requested from the problem. But besides this fact, there is also a more complicated reason, the algorithm is more complicated therefore to be taken in consideration. However, the main idea is always the same: to minimize error, individualizing the hyperplane which maximizes the margin, keeping in mind that part of the error is tolerated.

2.2.2 k-Nearest Neighbors

The k-nearest neighbours (k-NN) are supervised machine learning algorithms which can be used for classification and regression tasks [7]. The k-NN algorithm is one of

the simplest machine learning algorithms. It makes predictions for a new instance x by searching through the entire training set for the k most similar instances (the neighbours) and summarizing the output variable for those k instances. For regression this might be the mean output variable, in classification this might be the class value.

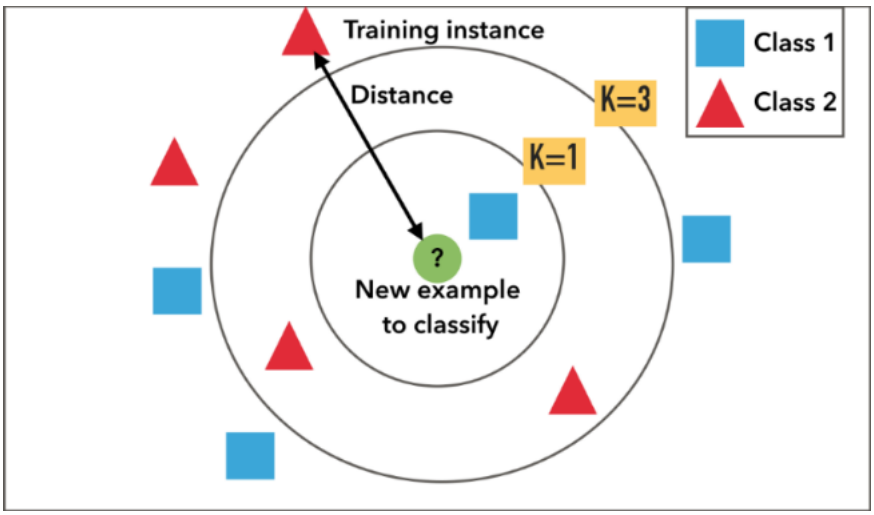


Figure 2.3: k-NN Illustration

More formally, given a training set with input sample features $x_1, x_2...x_n$ and the output result y_i, k-NN's goal is to learn a function $h : x \rightarrow y$, so that given an unseen sample x, $h(x)$ can confidently predict the corresponding output y.

The k-NN algorithm is also a non parametric and instance-based learning algorithm. Non-parametric means it makes no explicit assumptions about the functional form of h, avoiding the dangers of modeling incorrectly the underlying distribution of the data. Instance-based learning means that the algorithm does not

explicitly learn a model. Instead, it chooses to memorize the training instances which are subsequently used as knowledge for the prediction phase. Concretely, this means that only when a query to the database is made, will the algorithm use the training instances to spit out an answer.

As mentioned before, the k-nearest neighbor algorithm forms a majority vote between the k most similar instances to a given unseen observation. Similarity is defined according to a distance metric between two data points. A common choice is the Euclidean distance given by:

$$d(x, x') = \sqrt{(x_1 - x'_1)^2 + (x_2 - x'_2)^2 + ... + (x_n - x'_n)^2} \tag{2.13}$$

Other measures which can be more suitable for a given setting are the Minkovski, Chebyshev and Hamming distance.

For classification tasks, given a positive integer k, an unseen observation x and a similarity metric d, the k-NN classifier performs the following two steps:

- It runs through the whole data-set computing d between x and each training observation and defines the k points in the training data that are closest to x to set A.

- It then estimates the conditional probability for each class, which is the fraction of points in A with that given class label. Finally, assigns x to the class with the largest probability.

$$P(Y = j|X = x) = \frac{1}{k}\sum_{i \in A} I(y^{(i)} = j) \tag{2.14}$$

where $I(x)$ is the indicator function which evaluates to 1 when the argument x is true and 0 otherwise.

For regression tasks, the first step of the algorithm is setting A, as before. Although in this case, the predicted response for a new instance is the mean of the k nearest neighbors responses. Other summary statistics, such as the median, can also be used in place of the mean to predict the new sample.

2.2.3 Random Forest

2.2.3.1 Desicion Tree Algorithm

The decision tree algorithm belongs to the family of supervised learning algorithms and can be used for solving regression and classification problems, as well as for predictor importance estimation. There are various decision tree algorithms, namely ID3 (Iterative Dichotomiser 3), C4.5, CART (Classification and Regression Tree), CHAID (CHi- squared Automatic Interaction Detector), MARS. In this dissertation, the CART [8] algorithm is used for both classification and regression tasks. Furthermore, predictor importance estimation is accomplished using interaction test for splitting predictors in unbiased way.

The decision tree algorithm tries to solve the problem, by using tree repre-

sentation. Leaf node is an indicator of the value of an output feature (class). A decision node specifies all possible tests on a single input feature, with one branch and sub-tree for each possible outcome of the test [9].

Decision Tree Pseudo-code

1. Place the best feature of the data-set at the root of the tree.
2. Split the training set into subsets. Subsets should be made in such a way that each subset contains data with the same value for a feature.
3. Repeat step 1 and step 2 on each subset until you find leaf nodes in all the branches of the tree.

For predicting a class label for a record, the algorithm starts from the root of the tree and compares the values of the root feature with the record's feature. On the basis of comparison, it follows the branch corresponding to that value and jumps to the next node. It continues comparing the records feature values with other internal nodes of the tree until it reaches a leaf node with predicted class value.

2.2.3.2 Splitting Criterion

The primary challenge in the decision tree implementation is to identify which features are considered as the root node at each level. Handling this is known as feature selection. There are different feature selection measures to identify the feature which can be considered as the root node at each level.

For solving this feature selection problem, researchers worked and devised some solutions. They suggested using some criterion like information gain, Gini index, etc. These criterions calculate values for every feature. The values are sorted and the features are placed in the tree by following the order i.e, the feature with a high value is placed at the root.

Creating a decision tree is actually a process of dividing up the feature space. A greedy approach is used to divide the space called recursive binary splitting. This is a numerical procedure where all the values are lined up and different split points are tried and tested using a cost function. The split with the lowest cost is selected. All input variables and all possible split points are evaluated and chosen in a greedy manner based on the cost function.

For the standard CART algorithm:

- Regression: The cost function that is minimized to choose split points is the sum squared error across all training samples that fall within the rectangle:

$$\sum (y - prediction)^2 \tag{2.15}$$

- Classification: The Gini cost function is used which provides an indication of how pure the nodes are, where node purity refers to how mixed the training data assigned to each node is.

$$\sum (p_k \cdot (1 - p_k)) \tag{2.16}$$

Where G is the Gini index over all classes, p_k are the proportion of training instances with class k in the rectangle of interest. A node that has all classes of the same type (perfect class purity) will have G=0, where as a G that has a 50-50 split of classes for a binary classification problem (worst purity) will have a G=0.5.

Splitting continues until nodes contain a minimum number of training examples or a maximum tree depth is reached. The most common stopping procedure is to use a minimum count on the number of training instances assigned to each leaf node. If the count is less than some minimum then the split is not accepted and the node is taken as a final leaf node.

2.2.3.3 Random Forest

It is known that decision trees suffer from bias and variance. Simple trees have a large bias (underfitting) and complex trees have a large variance (overfitting). Ensemble methods combine several decision trees to produce better predictive performance than utilizing a single decision tree. The main principle behind the ensemble model is that a group of weak learners come together to form a strong learner. The most used techniques to perform ensemble of decision trees are bagging (bootstrap aggregation) and boosting.

Bagging is used when our goal is to reduce the variance of a decision tree, without increasing the bias. The idea is to create several subsets of the training data, chosen randomly with replacement. Now, each collection of subset data is used to train their decision trees. As a result, we end up with an ensemble of

different models. Average of all the predictions from different trees are used which is more robust than a single decision tree [10].

In this dissertation the bagging technique is used to create a random forest. Random forests differ in only one way from the general scheme of the bagging method: they use a modified tree learning algorithm that selects, at each candidate split in the learning process, a random subset of the input features. This process is sometimes called feature bagging. The reason for doing this is the correlation of the trees in an ordinary bagging sample. If one or a few features are very strong predictors for the response variable (target output), these features will be selected in many of the trees in the ensemble, causing them to become correlated. An analysis of how bagging and random subspace projection contribute to accuracy gains under different conditions is given by Ho [11].

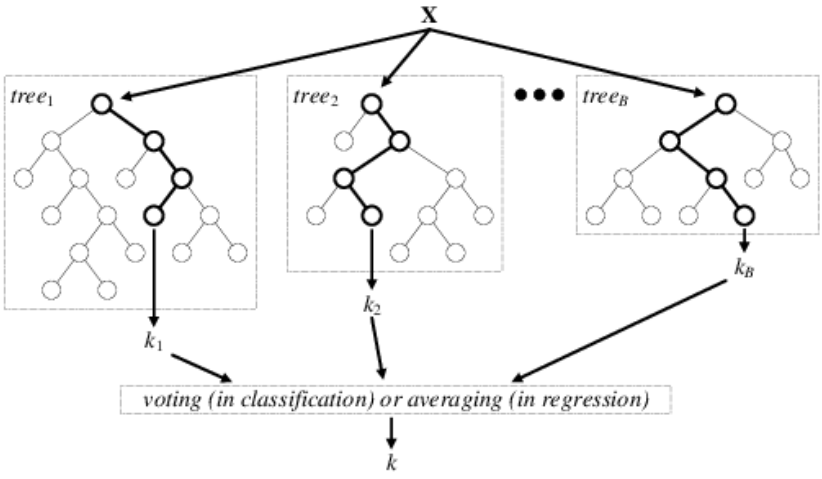


Figure 2.4: Random forest architecture

2.2.3.4 Out-of-bag Error

Each tree in a random forest is constructed based on a random sample of the observations. The observations that are not part of the sub-sample are referred to as out-of-bag observations. The out-of-bag observations can be used for estimating the prediction error of a random forest. The out-of-bag error is often used for assessing the prediction performance of a random forest. An advantage of the out-of-bag error is that the complete original data-set is used both for constructing the random forest model and for error estimation.

2.2.3.5 Predictor Importance Estimation

Out-of-bag, predictor importance estimates by permutation measure how influential the input features in the model are at predicting the response. The influence of a predictor increases with the value of this measure.

If a predictor is influential in prediction, then permuting its values should affect the model error. If a predictor is not influential, then permuting its values should have little to no effect on the model error.

The following process describes the estimation of out-of-bag predictor importance values by permutation. Suppose that R is a random forest of T learners and p is the number of predictors in the training data.

1. For a tree t, $t = 1, ..., T$:

    (a) Identify the out-of-bag observations and the indices of the predictor

variables that were split to grow tree t,$s_t \subseteq \{1, ..., p\}$.

(b) Estimate the OOB error ϵ_t.

(c) For each predictor variable x_i,$j \in s_t$

i. Randomly permute the observations of x_i.

ii. Estimate the model error, ϵ_{ti}, using the out-of-bag observations containing the permuted values of x_i.

iii. Take the difference $d_{ti} = \epsilon_{ti} - \epsilon_t$. Predictor variables not split when growing tree t are attributed a difference of 0.

2. For each predictor variable in the training data, compute the mean, $\bar{d}_i$ and standard deviation, σ_i, of the differences over the learners, $i = 1, ..., p$.

3. The out-of-bag predictor importance by permutation for x_i is $\bar{d}_i/\sigma_i$.

It is worth emphasizing that, in the case of predictor importance estimation the standard CART algorithm may give biased estimations, because it is not sensitive to predictor variable interactions. Standard CART tends to select split predictors containing many distinct values, e.g. continuous variables, over those containing few distinct values, e.g. categorical variables [12]. For unbiased predictor importance estimation, the interaction test is used rather than the standard CART algorithm.

The interaction test chooses the split predictor that minimizes the p-value of *chi*-square tests of independence between each predictor and the response, and that minimizes the p-value of a *chi*-square test of independence between each pair

of predictors and response [12]. Usually, the training speed is slower than standard CART.

2.2.4 Gaussian Process Regression

In probability theory and statistics, a Gaussian process is a stochastic process (a collection of random variables indexed by time or space), such that every finite collection of those random variables has a multivariate normal distribution, i.e. every finite linear combination of them is normally distributed. The distribution of a Gaussian process is the joint distribution of all those (infinitely many) random variables, and as such, it is a distribution over functions with a continuous domain, e.g. time or space.

A machine-learning algorithm that involves a Gaussian process uses lazy learning and a measure of the similarity between points (the kernel function) to predict the value for an unseen point from training data. The prediction is not just an estimate for that point, but also has uncertainty information. It is a one-dimensional Gaussian distribution (which is the marginal distribution at that point) [13].

Consider a training set $(x_i, y_i), i = 1, 2, ..., n$, where $x_i \in \Re^d$, drawn from an unknown distribution. A Gaussian process regression (GPR) model addresses the question of predicting the value of a response variable y_{new}, given the new input vector x_{new} and the training data [14]. A linear regression model is of the form

$$y = x^T \beta + \epsilon \tag{2.17}$$

where $\epsilon \sim \mathcal{N}(0, \sigma_2)$. The error variance σ_2 and the coefficients β are estimated from the data. A Gaussian process regression model explains the response by introducing latent variables, $f(x_i)$,i=1,2,...,n, from a Gaussian process (GP) and explicit basis functions, h. The kernel (or covariance) function of the latent variables captures the smoothness of the response and basis functions project the inputs x into a p-dimensional feature space.

A GP is a set of random variables, such that any finite number of them have a multivariate Gaussian distribution. If $f(x), x \in \Re^d$ is a GP, then given n observations $x_1, x_2, .., x_n$, the multivariate distribution of the random variables $f(x_1), f(x_2), ..., f(x_n)$ is Gaussian. A GP is defined by its mean function $m(x)$ and covariance function, $k(x, x')$. That is, if $f(x), x \in \Re^d$ is a Gaussian process, then $E(f(x)) = m(x)$ and $Cov[f(x), f(x')] = E[\{f(x) - m(x)\}\{f(x') - m(x')\}] = k(x, x')$.

The GPR model is represented as:

$$h(x)^T\beta + f(x) \tag{2.18}$$

where $f(x) \sim \mathcal{GP}(0, k(x, x'))$, that is $f(x)$ are from a zero mean GP with covariance function, $k(x, x')$. $h(x)$ are a set of basis functions that transform the original feature vector x in $\Re^d$ into a new feature vector $h(x)$ in $\Re^p$. β is a p-by-1 vector of basis function coefficients. An instance of response y can be modeled as :

$$P(y_i \mid f(x_i), x_i) \sim \mathcal{N}(y_i \mid h(x_i)^T\beta + f(x_i), \sigma^2) \tag{2.19}$$

Hence, a GPR model is a probabilistic model. There is a latent variable $f(x_i)$ introduced for each observation x_i, which makes the GPR model non-parametric. In vector form, this model is equivalent to

$$P(y \mid f, X) \sim \mathcal{N}(y \mid H\beta + f, \sigma^2 I) \tag{2.20}$$

The joint distribution of latent variables $f(x_1), f(x_2), ..., f(x_n)$ in the GPR model is as follows:

$$P(f \mid X) \sim \mathcal{N}(f \mid 0, K(X, X)) \tag{2.21}$$

The kernel function $k(x, x')$ is usually parameterized by a set of kernel hyper-parameters, θ. Often $k(x, x')$ is written as $k(x, x' \mid \theta)$ to explicitly indicate the dependence on θ.

For many standard kernel functions, the kernel parameters are based on the signal standard deviation σ_f and the characteristic length scale σ_l. The characteristic length scales briefly define how far apart the input values x_i can be for the response values to become uncorrelated. Both σ_l and σ_f need to be greater than 0, and this can be enforced by the unconstrained parametrization vector , such that $\theta_1 = \log \sigma_l$ and $\theta_2 = \log \sigma_f$.

Some of the commonly used GPR kernel functions are:

- Linear

$$k(x, x') = x^T x'$$

- Exponential

$$k(x, x') = \sigma_f^2 \exp(-\frac{r}{\sigma_l})$$

- Matern 5/2

$$k(x, x') = \sigma_f^2(1 + \frac{\sqrt{5}r}{\sigma_l} + \frac{\sqrt{5}r^2}{\sigma_l} \exp -\frac{\sqrt{5}r}{\sigma_l})$$

- Rational Quadratic

$$k(x, x') = \sigma_f^2(1 + \frac{r^2}{2\alpha\sigma_l^2})^{-a}$$

where, r is the Euclidean distance between x and x'.

While the basis function can be one of the below functions:

- Constant $H = 1$
- Linear $H = [1, X]$
- Pure Quadratic $H = [1, X, X_2]$

2.3 Bias and Variance in Machine Learning Models

The performance of a machine learning model is considered good based on its predictions and how well it generalizes on an unseen test data-set. The bias-variance trade-off is a way of analyzing the expected generalization error of a learning algorithm, with respect to a particular problem as a sum of three terms, the bias, variance, and a quantity called the irreducible error, resulting from noise in the problem itself. It can be applied to all forms of supervised learning [15].

The bias is an error from erroneous assumptions in the learning algorithm. In other words, it is the difference between the average prediction of the model and the correct target value. Model with high bias pays very little attention to the training data and oversimplifies the model. It always leads to high error on training and test data.

The variance is an error from sensitivity to small fluctuations in the training set. High variance can cause an algorithm to model the random noise in the training data, rather than the intended outputs. Model with high variance pays a lot of attention to training data and does not generalize on unseen data. As a result, such models perform very well on training data but has high error rates on test data.

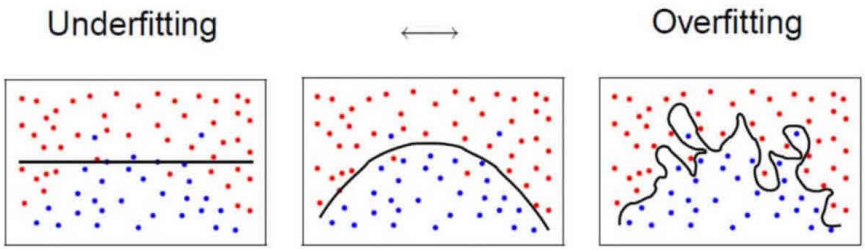


Figure 2.5: Overfitting, Underfitting, Right Balance

In supervised learning, underfitting happens when a model is unable to capture the underlying pattern of the data. These models usually have high bias and low variance. It happens when less amount of data is available to build an accurate model or when a linear model a applied to nonlinear data.

On the other hand, overfitting happens when a model captures the noise along

with the underlying pattern in data. It happens when a model is trained over noisy data-set. These models have low bias and high variance.

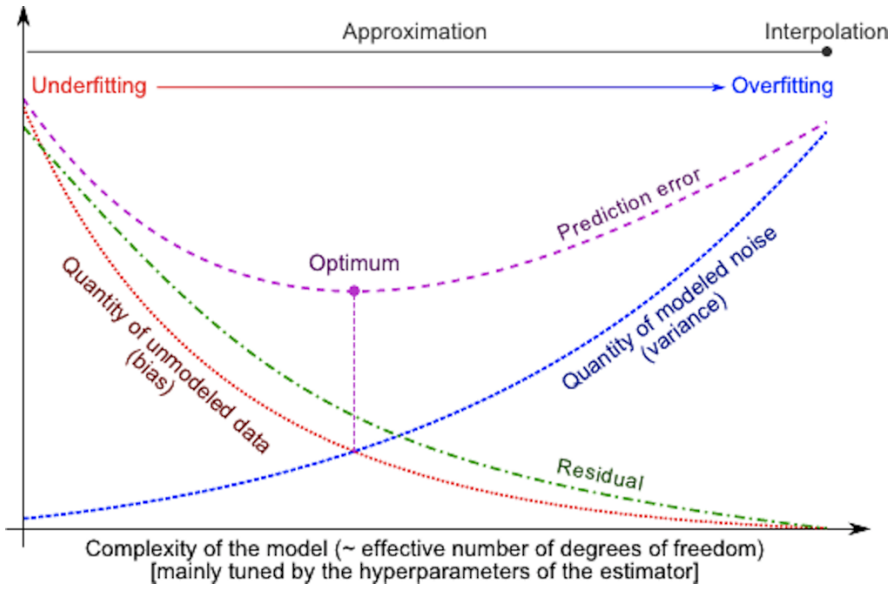


Figure 2.6: Optimum model complexity

If a model is too simple and has very few parameters then it may have high bias and low variance. On the other hand, if a model has large number of parameters then it is going to have high variance and low bias. This trade-off in complexity is why there is a trade-off between bias and variance. An algorithm cannot be more complex and less complex at the same time. To build a good model, there is a need to find a good balance between bias and variance such that it minimizes the total error, as defined below.

$$Total\ Error = Bias^2 + Variance + Irreducible\ Error \tag{2.22}$$

2.4 Cross-Validation

As mentioned above, to obtain a model that generalizes well on unseen data, there is a need to minimize the total error finding a good balance between bias and variance. Hence, there is a need to find a way to accurately measure the prediction error of the model. A common approach is using the data itself to estimate the true prediction error.

The simplest technique of this approach is the holdout set method. Here the data is initially split into two groups. One group will be used to train the model, the second group will be used to measure the resulting model's error.

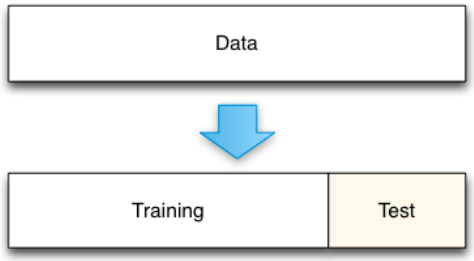


Figure 2.7: Holdout data split

The cost of the holdout method comes in the amount of data that is removed from the model training process. Training the model on a smaller data usually gives a higher prediction error (higher bias). As a solution, a re-sampling based technique such as cross-validation will be used instead.

Cross-validation works by splitting the data into a set of n folds. The model building and error estimation process is repeated n times. Each time $n-1$ groups

are combined and used to train the model. The group that was not used to construct the model is used to estimate the true prediction error. After this process, we end up with n error estimates that are averaged to obtain a more robust estimate of the true prediction error.

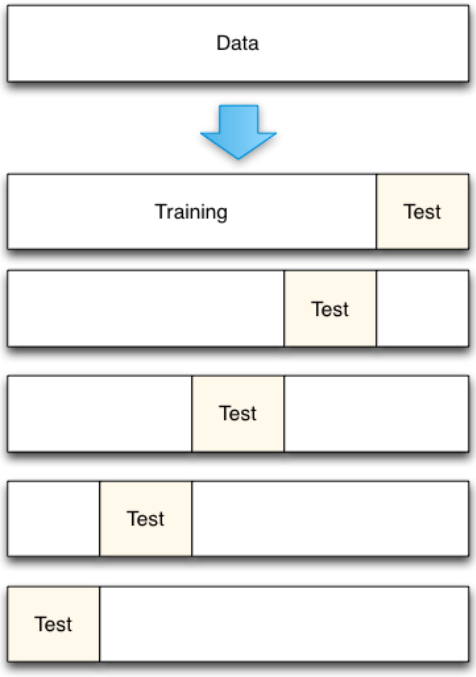


Figure 2.8: 5-fold cross-validation data split

As can be seen, cross-validation is very similar to the holdout method. Where it differs, is that each data point is used both to train models and to test a model, but never at the same time. Cross-validation can also give estimates of the variability of the true error estimation which is a useful feature.

In this dissertation, the 5-fold cross-validation method is used to assess the

performance of the SVM, k-NN, RF and GPR models. It is worth noting that, the cross-validation loss is also used as the objective function within the Bayesian optimization, which is discussed in the next section.

2.5 Hyper-parameter Optimization

The aim of hyper-parameter optimization in machine learning is to find the hyper-parameters of a given machine learning algorithm that return the best performance as measured on a validation set. In other words, hyper-parameter optimization finds a group of hyper-parameters that yields an optimal model which minimizes a predefined loss function on given independent data.

Hyper-parameters, in contrast to model parameters, are set by the model developer before training and they control the learning process. The number of trees in a random forest and the penalization parameter C in a SVM are hyper-parameters while the weights in a SVM are model parameters learned during training.

The problem with manual hyper-parameter optimization is that evaluating the objective function to find the score is extremely expensive. Each time different hyper-parameters are tried, a model on the training data is trained to make predictions on the validation data and then the validation metric is calculated. With a large number of hyper-parameters and complex models this process quickly becomes intractable to do by hand.

Grid search and random search are slightly better than manual tuning because a grid of model hyper-parameters is set up and the train-predict-evaluate cycle

runs automatically in a loop. However, even these methods are relatively inefficient because they do not choose the next hyper-parameters to evaluate based on previous results. Grid and random search are completely uninformed by past evaluations.

Bayesian approaches, in contrast to random or grid search, keep track of past evaluation results which they use to form a probabilistic model mapping hyper-parameters to a probability of a score on the objective function $P(score \mid hyperparameters)$.

In the literature, this model is called a surrogate for the objective function. The surrogate is much easier to optimize than the objective function and Bayesian methods work by finding the next set of hyper-parameters to evaluate on the actual objective function by selecting hyper-parameters that perform best on the surrogate function $f(x)$.

Bayesian Optimization uses a Gaussian Process to fit the surrogate model $f(x)$, as described in section 2.2.4. One innovation in Bayesian optimization is the use of an acquisition function, which the algorithm uses to determine the next point to evaluate. The acquisition function can balance sampling at points that have low modeled objective functions and exploring areas that have not yet been modeled well.

The algorithm evaluates $y_i = f(x_i)$ for n points x_i, taken at random within the variable bounds. If there are evaluation errors, it takes more random points until there are n successful evaluations. The probability distribution of each component is either uniform or log-scaled.

Then it repeats the following steps:

1. Updates the Gaussian process model of $f(x)$ to obtain a posterior distribution over functions $Q(f \mid x_i, y_i \text{ for } i = 1, ..., t)$.

2. Finds the new point x that maximizes the acquisition function $a(x)$.

The algorithm stops after reaching a fixed number of iterations or a fixed time or a stopping criterion.

The acquisition function evaluates the goodness of a point x based on the posterior distribution function Q. Then it selects the point with the lowest expected loss. Some of the acquisition functions that can bu used in Bayesian optimization is expected improvement, probability of improvement and lower confidence bound.

In this dissertation, Bayesian hyper-parameter optimization is used with expected improvement acquisition function. The expected improvement acquisition functions evaluates the expected amount of improvement in the objective function, ignoring values that cause an increase in the objective. In other words, it defines x_{best} as the location of the lowest posterior mean and $\mu_Q(x_{best})$ as the lowest value of the posterior mean. Then the expected improvement is $EI(x, Q) = E_Q[max(0, \mu_Q(x_{best}) - f(x))]$.

Bayesian hyper-parameter optimization finds the optimum hyper-parameters which minimize the cross-validation loss, following the above procedure. Then, these objective function evaluations, namely the optimum model hyper-parameters are used to train a new cross-validated model.

2.6 Performance Evaluation Metrics

As mentioned in section 2.4, the k-fold cross-validation method is used to assess the performance of the trained models. In case of classification, the model evaluation metric is the classification accuracy. While in regression problems the root mean square error is used as model evaluation metric. In this section, another performance metrics which can be applied to regression and classification problems are presented.

Classification

Classification accuracy is the easiest classification metric to understand. But, it does not give any information about the underlying distribution of response values and the "types" of errors the classifier is making. Thus, for the classification models the confusion matrix is also calculated.

A confusion matrix shows the number of correct and incorrect predictions made by the classification model compared to the actual outcomes (target value) in the data. The matrix is NxN, where N is the number of target values (classes). Performance of such models is commonly evaluated using the data in the matrix. Figure 2.9 displays a 2x2 confusion matrix for two classes (Positive and Negative).

- Accuracy : the proportion of the total number of predictions that were correct.

- Positive Predictive Value or Precision : the proportion of positive cases that were correctly identified.

Confusion Matrix		Target			
		Positive	Negative		
Model	Positive	a	b	*Positive Predictive Value*	a/(a+b)
	Negative	c	d	*Negative Predictive Value*	d/(c+d)
		Sensitivity	*Specificity*	**Accuracy** = (a+d)/(a+b+c+d)	
		a/(a+c)	d/(b+d)		

Figure 2.9: Confusion Matrix for Binary Classification Model

- Negative Predictive Value : the proportion of negative cases that were correctly identified.
- Sensitivity or Recall : the proportion of actual positive cases which are correctly identified.
- Specificity : the proportion of actual negative cases which are correctly identified.

Regression

The error rate of the regression models is measured with the root mean square error, which is given

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(p_i - a_i)^2}{n}} \tag{2.23}$$

where a is the actual target, p is the predicted target and n is the number of samples.

Besides the root mean square error, the predicted vs. actual plot and the residuals plot are also used in order to visualize the results of a regression model.

The predicted vs. actual shows how well the regression model makes predictions for different response values. The predicted response of the model is plotted against the actual, true response. A perfect regression model has a predicted response equal to the true response, so all the points lie on a diagonal line. The vertical distance from the line to any point is the error of the prediction for that point. A good model has small errors, and so the predictions are scattered near the line.

The residuals plot displays the difference between the predicted and true responses. Usually a good model has residuals scattered roughly symmetrically around zero. If any clear patterns in the residuals exists, it is likely that the model needs improvement.

Chapter 3: Rocking

3.1 Presentation of the problem

Reconnaissance reports following strong earthquakes substantiate that a solitary free-standing solid body subjected to a seismic excitation of the base can uplift, slide, rock or overturn. The need to understand and predict these failures in association with the temptation to estimate levels of ground motion by examining whether slender structures have overturned or survived the earthquakes, has motivated a number of studies on the response of blocks.

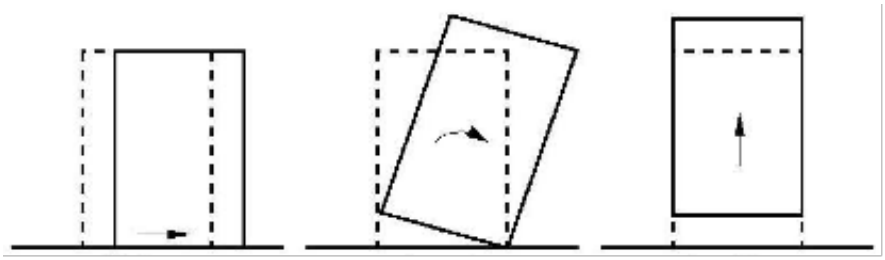


Figure 3.1: Seismic Response of a solitary free-standing solid body under a seismic excitation

The phenomenon of the partial uplift of a structure from its foundation and its oscillation when the center of rotation changes simultaneously from one point of reference to another is known in the literature as rocking and it is often observed during seismic excitation. The study of rocking motion in structures is

crucial. As it was indicated by many researchers, slender structures may rock on their foundation, a phenomenon which can be devastating in some cases. Rocking motion introduces a highly nonlinear mechanical problem due to the fact that it involves a wide range of nonlinear physical phenomena, such as impact, contact, uplift and sliding. The rocking problem can either be modelled through nonlinear finite element analysis or analytical solutions can be applied to solve the equation of motion of the rocking block.

3.2 Sliding of Solitary Blocks - Coefficient of friction

During the dynamic movement of a free-standing solid body due to the base excitation, a nonlinear phenomenon which may take place is sliding. A body slides if it can not follow the ground motion. The friction in the interface between the base and the body has a direction parallel to the interface and is produced by the relative displacement of the two surfaces in contact and does not depend on their sliding velocities. If the horizontal force exceeds the static force of friction (which is a boundary value) then the body is sliding. The ratio of the static force of friction to the vertical base resistance is called static coefficient of friction:

$$\mu_{st} = \frac{T_{st}}{N} \tag{3.1}$$

During the sliding the body continues to exist forces of friction. The requirement for continuation of sliding with fixed velocity is a horizontal force called

sliding force of friction and is constant during the sliding of the body. The ratio of the sliding force of friction to the vertical base resistance is called sliding coefficient of friction:

$$\mu_{tot} = \frac{T_{tot}}{N} \tag{3.2}$$

It is generally accepted that $\mu_{tot} < \mu_{st}$. The static coefficient of friction and the sliding coefficient of friction is independent of the body mass and the area of the contact surface and depends only from the kind of the contact surfaces.

Assuming that $\ddot{V}$ and $\ddot{U}$ are the vertical and horizontal base accelerations respectively, X and Y are the horizontal and vertical displacements, $\dot{X}$ and $\dot{Y}$ are the velocities and $\ddot{X}$ and $\ddot{Y}$ are the horizontal and vertical accelerations respectively of the body.

According to the principle of D 'Alembert the equations of motion of the body will be:

$$m\ddot{X} = T \tag{3.3}$$

$$m\ddot{Y} = N - mg \tag{3.4}$$

where m is the body mass. Modifying the above equations we have as a result:

$$T = m\ddot{U} \tag{3.5}$$

$$N = mg\Big(1 + \frac{\ddot{V}}{g}\Big) \tag{3.6}$$

The inertial force of the free-standing solitary body must be at least the same with the static force of friction, in order to slide:

$$m\ddot{X} \geq T_{st} \tag{3.7}$$

Taking into account all the above is arising:

$$\mu\ddot{X} \geq T_{st} \Rightarrow$$

$$\mu\ddot{X} \geq \mu_{st}N \Rightarrow$$

$$\mu\ddot{X} \geq \mu_{st}mg\Big(1 + \frac{\ddot{V}}{g}\Big) \Rightarrow$$

$$\ddot{U} \geq \mu_{st}g\Big(1 + \frac{\ddot{V}}{g}\Big)$$

Ignoring the vertical ground motion (the vertical acceleration) then sliding takes place only in case of:

$$\ddot{U} \geq \mu_{st}g \tag{3.8}$$

3.3 Rocking Motion of Solitary Blocks

As it is described above the phenomenon of the partial uplift of a structure from its foundation and its oscillation when the center of rotation changes simultaneously from one point of reference to another is known in the literature as rocking.

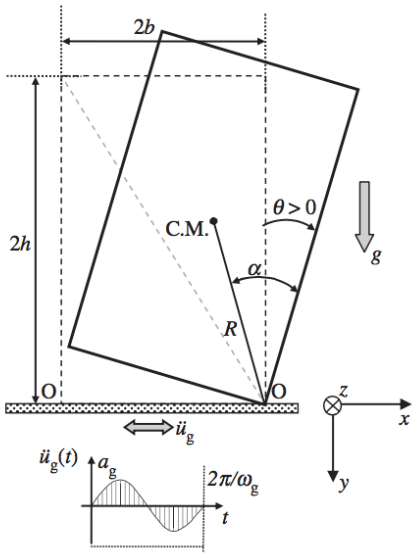


Figure 3.2: Characteristics of rocking block

It has been proved that rocking initiates when during the excitation :

$$\ddot{u}_g \geq \alpha_{g,min} = g tan\alpha \tag{3.9}$$

where $tan\alpha = h/b$ and $2h$, $2b$ are the height and width of a rectangular body.

In this masters dissertation it is assumed that there is no sliding. This assumption means that $tan\alpha < \mu_{st}$ which is acceptable in the majority of structures which are significant for a Civil Engineer. If $tan\alpha > \mu_{st}$ then the body is not rocking and consequently is sliding. Structures with $tan\alpha > \mu_{st}$ are not examined here.

The rocking motion is usual in free-standing body and generally in free-standing structures due to the above observation. For this reason, for more than a century the response of rigid blocks allowed to uplift and rock on a rigid foundation under

seismic ground motion excitation has been studied. Housner [16] demonstrated that large rigid blocks and large rocking rigid frames is difficult to overturn dynamically. In the same way Housner has shown a scale effect that characterizes the response of rocking blocks subjected to a ground motion.

For a given earthquake, larger objects need a larger ground acceleration to overturn and longer dominant period earthquakes have a larger overturning capability that shorter dominant period ones. This explains the survival of ancient Greek and Roman top-heavy temple structures in regions of high seismicity, despite the lack of historical evidence that ancient engineers were aware of the size effect of rocking structures. This size effect has lead researchers to propose rocking as a seismic response modification technique. A 60-m-tall bridge designed to rock has already been built across the Rangitikei River in New Zealand in 1981. Moreover, a 33-m-tall chimney at the Christchurch New Zealand airport has been designed to uplift. Furthermore, three 30 to 38-m-tall chimneys in Piraeus, Greece, have been retrofitted by letting them uplift in case of an earthquake.

3.4 Equation of Motion of Damped Rigid Block

Consider a rectangular block with height $2h$ and width $2b$, as shown in figure 3.2. If the coefficient of friction between the block and the base is infinite so that there is no sliding, the equation of motion of a solitary free standing block with $R = \sqrt{h^2 + b^2}$ and slenderness $\alpha = \arctan(b \backslash h)$ under a horizontal ground acceleration $\ddot{u}_g(t)$, when rocking around the pivot points O and O' respectively is

[16–19]:

$$I_0\ddot{\theta}(t) + mgRsin[\alpha sgn(\theta(t)) - \theta(t)] = -m\ddot{u}_g Rcos[\alpha sgn(\theta(t)) - \theta(t)] \tag{3.10}$$

where $I_0 = (4\backslash 3)mR^2$ is the moment of inertia with respect to the pivot point, g is the gravity acceleration, m is the mass of the block and θ is the response rotation.

Rocking initiates when during the excitation occurs $\ddot{u}_g \geq \alpha_{g,min} = gtan\alpha$. Replacing I_0 to the above equation and defining the quantity $p = \sqrt{\frac{3g}{4R}}$ as the frequency parameter of the rocking block, equation takes the following form:

$$\ddot{\theta}(t) = -p^2\Big[sin[\alpha sgn(\theta(t)) - \theta(t)] + \frac{\ddot{u}_g}{g}cos[\alpha sgn(\theta(t)) - \theta(t)]\Big] \tag{3.11}$$

Assuming that θ and α are small or linearizing the equation of motion, it can be expressed in a simplified form as:

$$I_0\ddot{\theta}(t) + mgRsin[\alpha sgn\theta - \theta] = -m\ddot{u}_g R \tag{3.12}$$

Energy dissipation in rocking bodies takes place instantaneously at each impact, when the rotation changes sign at $\theta = 0$. The per-cycle of free vibration energy dissipation for a rigid rectangular block is described by the restitution factor r and it is independent of the amplitude of vibration. The ratio of the energy after one complete cycle, E, to the initial energy, E_0 is:

$$\frac{E}{E_0} = r^2 = \Big(1 - \frac{3}{2}\sin^2\alpha\Big)^4 \tag{3.13}$$

In most cases, impact is described by a resulting coefficient of restitution that relates the post-impact angular velocity $\dot{\theta}^+$ to the pre-impact angular $\dot{\theta}^-$ velocity and is:

$$\eta = \frac{\dot{\theta}^+}{\dot{\theta}^-} = 1 - \frac{3}{2}\sin^2\alpha \tag{3.14}$$

The solution of the equations is obtained numerically via a state-space formulation with standard ODE23s solver available in MATLAB.

3.5 Rocking Response Under One Sine-Pulse

The acceleration, velocity and displacement histories of a rigid block under one sine-pulse excitation, according to Zhang and Makris [20], are presented in figure 3.3.

Under the minimum acceleration amplitude, blocks overturn during their free-vibration regime at a theoretically infinite large time when the velocity tends to reach a local minimum [21]. Accordingly the condition for overturning is that

$$\ddot{\theta}(t_\infty) = 0 \tag{3.15}$$

where t_∞ = sufficiently large time, where $tanh(pt_\infty) = 1$.

Under a one-sine pulse, a free-standing block has two modes of overturning: (1) overturning with one impact (mode 1); and (2) overturning with no impact (mode 2). This result is true as long as $\frac{\omega_p}{p}$ is sufficiently small. As $\frac{\omega_p}{p}$ increases, the first mode of impact vanishes and the block overturns only without impact (mode

2). Accordingly, to back-figure the minimum overturning acceleration amplitude by imposing the condition of overturning given by 3.15, one has to distinguish between modes 1 and 2.

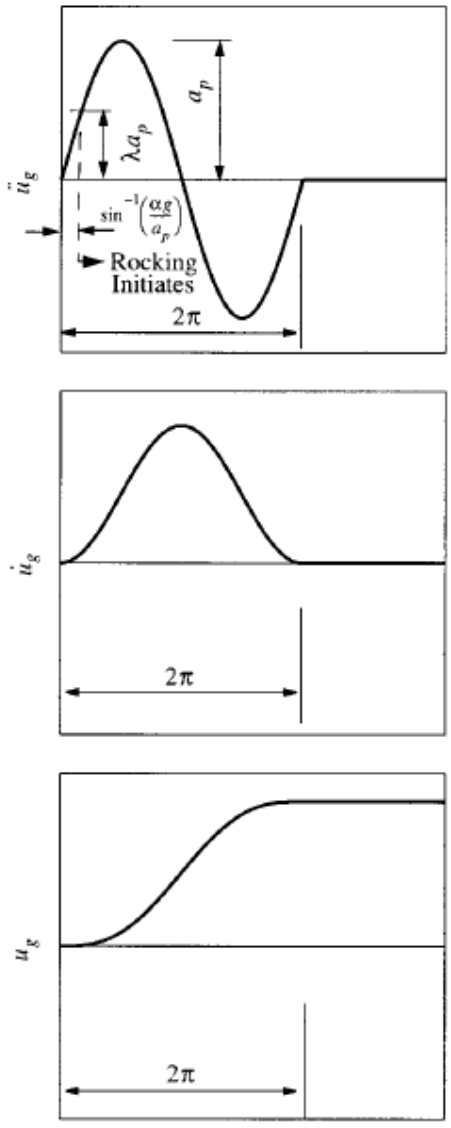


Figure 3.3: Acceleration, velocity, and displacement histories of one-sine pulse

3.5.1 Overturning Mode 1

Denoting the time as t_{fv} when the block enters its free-vibration regime, the condition for overturning after the block has experienced one impact (mode 1) is

$$\dot{\theta}(t_{fv}) + p[\theta(t_{fv}) - \alpha] = 0 \tag{3.16}$$

According to overturning mode 1:

Case 1 : Impact happens before the excitation expires ($t_i < T_{ex}$), $t_f v = T_{ex} = (2\pi - \psi)\frac{\omega_p}{p}$. The solution of the related analytical relations gives the minimum overturning moment of the block. Within the limits of the linear approximation (slender block) and assuming a value of $\eta = 0.9$, this happens when $0 \leq \frac{\omega_p}{p} \leq 4.8$.

Case 2 : Impact happens after the excitation expires ($t_i > T_{ex}$). The solution of the related analytical relations gives the minimum overturning moment of the block.

3.5.2 Overturning Mode 2

Under this mode, the block does not experience any impact. The condition of overturning becomes

$$\frac{\dot{\theta}(T_{ex})}{p} + [\theta(T_{ex}) + \alpha] = 0 \tag{3.17}$$

As in the previous cases, the solution of the arising analytical relations gives the minimum acceleration amplitude that is capable of overturning the block without

any impact.

Fig. 3.4 plots the solutions of the condition of overturning (for $\eta = 0.9$) after distinguishing carefully between modes 1 and 2 of overturning. When $\frac{\omega_p}{p}$ is sufficiently small, the minimum overturning acceleration is the result of mode 1. Overturning with mode 2 may also happen; however, a much higher acceleration amplitude is needed to manifest it. The distinction between modes 1 and 2 of overturning is of particular interest, because the transition from overturning with one impact to overturning without impact is not immediate and there is a finite margin of acceleration amplitudes with magnitudes larger than the minimum overturning acceleration (that corresponds to mode 1) that are unable to overturn the block.

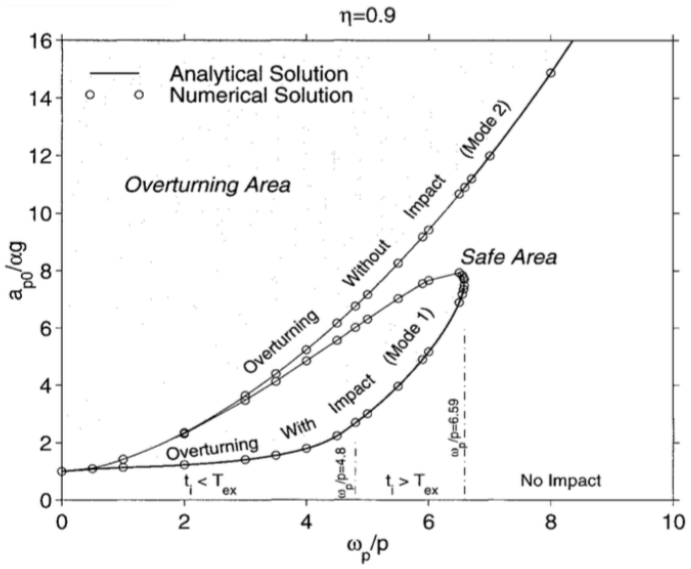


Figure 3.4: Overturning acceleration spectrum of free-standing block with $\eta = 0.9$ subjected to one-sine acceleration pulse with frequency ω_p

Closed-form solutions that define completely the overturning areas of the linearized rocking block under sine pulse excitation are derived from Dimitrakopoulos and De Jong [19].

Chapter 4: Application of Machine Learning Algorithms to Rocking Problems

The theory behind the machine learning algorithms, the hyper-parameter tuning as well as their performance evaluation are discussed in chapter 2. An analytic method to solve the equation of motion of a damped rigid block, as well as it's response under one sine-pulse excitation are presented in chapter 3. In this chapter, this knowledge is combined to create a training data-set and to train several machine learning models to make accurate predictions for new blocks.

Since a machine learning model with low generalization error is trained, the response of thousands rigid blocks can be estimated in a few seconds. A well trained model can be very useful, at least for a quick initial control, helping us to avoid computationally expensive tasks. The considered models, the preparation of the training data, the implementation of the models using MATLAB and their performance comparison are presented in the next sections.

Specifically, a training set which consists of rigid block samples under one sine base excitation is created and used for classification and regression models. support vector machine, k-nearest neighbours and random forest models are used for classification and their performance are compared. For regression, support vector regression, random forest and Gaussian process regression models are trained and their performance are compared.

4.1 Training Data Preparation and ML models

4.1.1 Training Data Preparation

The aim of the models which are presented in this chapter is to predict accurately the response of rigid blocks subjected to one sine-pulse base excitation and to classify accurately their overturning condition as described at [20], [19]. Thus, the training set consists of rigid block samples subjected to sine pulse base excitation. The equation of motion of the rocking block is solved using ODE23s solver available in MATLAB and the response rotation of each block is calculated.

The target output is the block's rotation over the slenderness angle, $\frac{\theta}{\alpha}$. In the case of rigid blocks under ground excitation, the only degree of freedom which is considered is the rotation of the block (rocking without uplifting and sliding). Hence, the ratio of the block's rotation over the slenderness angle is a representative output parameter for the models.

Besides the output, the accuracy of the predictions of the model depends strongly on the input features and how well they describe the motion of the block and the characteristics of the ground motion. The input features can vary depending on the examined problem.

In this case, sine pulse characteristics such as, the excitation frequency ω_p, amplitude of the acceleration A_g and block characteristics such as, its characteristic frequency p can be used as input parameters in the training data-set. In order to visualize the data-set and the model results as their are presented in [20] and to be able to predict the overturning conditions as described from J. Zhang and N.

Makris, Fig. 3.4, $\frac{\omega_p}{p}$ and $\frac{A_g}{gtan\alpha}$ are used as input features of the training set.

Totally 1242 rocking block instances are solved for A_g=1 ÷ 12 gtanα , T_g=0.2 ÷ 1.5 sec and for $R = 0.5m$ and $R = 1m$ and $\eta = 0.85$. The block's rotation ratio over the slenderness angle is calculated for each block.

4.1.2 Models

The training data-set which is described above is used to train classification and regression models. In regression, the algorithm is trained with the corresponding input features and the output result $\frac{\theta}{\alpha}$. The aim of the model is to predict accurately the response $\frac{\theta}{\alpha}$ for new block instances, given the input features. In the rocking block problem this response prediction has content for $\frac{\theta}{\alpha} < 1$, considering that for values $\frac{\theta}{\alpha} \geq 1$ the block overturns.

Classification models are also trained with the same input features, but they try to predict a category in which an instance belongs. For the rocking problem, the rigid blocks which are subjected to base excitation can be categorized as overturning blocks and safe blocks according to their response. This categorization forms a binary classification problem. Specifically, the block rocks safely for $\frac{\theta}{\alpha} < 1$ and overturns for $\frac{\theta}{\alpha} \geq 1$. Therefore, each block sample is labelled as "*overturned*" or "$safe$". Furthermore, they can be categorized as blocks that overturn with impact, blocks that overturn without impact and blocks that remain safe. This forms a three class classification problem. In the three class classification problem, for $\frac{\theta}{\alpha} \geq 1$ and impacts=0 the block sample is labeled as "*overturned*", for $\frac{\theta}{\alpha} \geq 1$ and

impacts >1 the block sample is labeled as "$overturned - impact$" and for $\frac{\theta}{\alpha} < 1$ the sample is labeled as "$safe$".

The training data-sets for regression, for binary and three class classification are presented below in tables 4.1, 4.2 and 4.3.

Table 4.1: Regression training set properties

Input Features	$\frac{\omega_p}{p}$	$\frac{A_p}{gtan\alpha}$
Output Labels	$\frac{\theta}{a}$	
Number of Samples	213	

Table 4.2: Binary classification training set properties

Input Features	$\frac{\omega_p}{p}$	$\frac{A_p}{gtan\alpha}$
Output Labels	"$safe$"	"$overturned$"
Number of Samples	1242	

Table 4.3: Three class classification training set properties

Input Features	$\frac{\omega_p}{p}$	$\frac{A_p}{gtan\alpha}$	
Output Labels	"$overturned$"	"$overturned - impact$"	"$safe$"
Number of Samples	1242		

4.2 Classification Models

4.2.1 Support Vector Machine

Binary Classification

MATLAB's *fitcsvm* function trains or cross-validates a SVM model for binary classification on a low-dimensional or moderate-dimensional predictor data-set. Model properties, such as the kernel function, kernel hyper-parameters, hyper-parameter optimization and cross-validation are defined as name-value pair arguments within the function.

First of all, the suitable kernel function for the training data-set have to be chosen. For linearly separable data the linear kernel would be a good choice. In our case, both training data-sets contain linearly inseparable data. Thus, Gaussian and polynomial kernels are better options. It is worth mentioning that the input features must be scaled when using kernel functions. Because, if the features do not have comparable ranges, the features with the largest range will completely dominate in the computation of the kernel matrix.

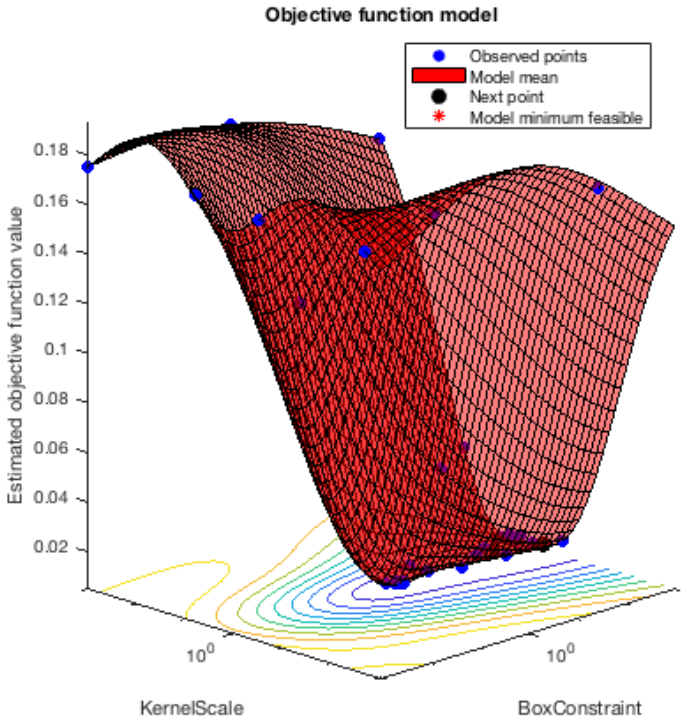
The optimum kernel hyper-parameters are defined using Bayesian optimization. An initial model is constructed, defining the Gaussian kernel function and using 5-fold cross-validation to calculate the classification loss. The eligible optimization parameters are the penalty parameter C and σ^2 and the objective function is the cross-validation loss. The optimization process is presented in Fig. 4.1. The optimum values for C ans σ^2 are 37.7 and 2.2 respectively. While the cross-validation loss of the model for this combination is 0.004.

After the determination of the optimum model hyper-parameters, the final model is built and used to make predictions on new data. The generalization error of the model is defined with 5-fold cross-validation.

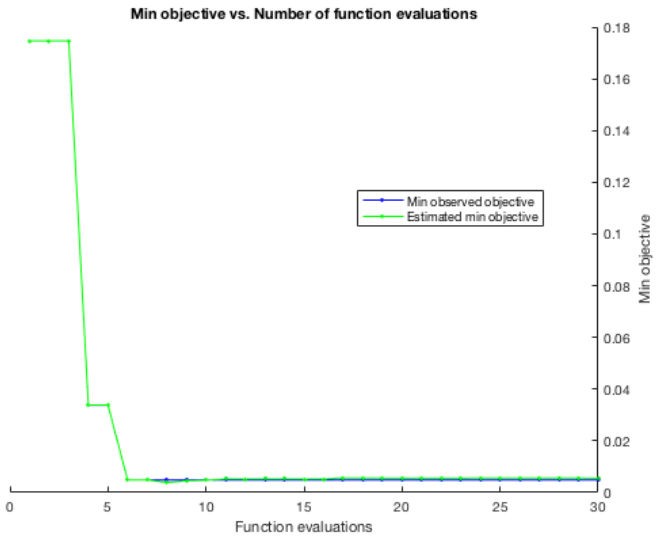
After the training and the estimation of the generalization error, MATLAB's *predict* function is used to make predictions on unseen input pairs with the trained SVM model. Specifically, a mesh grid within the input feature range is created and the predictions of the algorithm on this grid are plotted along with the training data points. In this way, the decision plane of the algorithm can be compared with the training set in order to evaluate the performance of the classifier, Fig. 4.3. Another performance evaluation metric is the confusion matrix, Fig. 4.2.

Listing 4.1: Binary classification with support vector machines, training and cross-validation

```
%BINARY SVM TRAINING
classificationSVM = fitcsvm(...
predictors, ... %input
response, ...   %output
'KernelFunction', 'gaussian', ...
'PolynomialOrder', [], ...
'KernelScale', 2.22, ...
'BoxConstraint', 37.7, ...
'Standardize', true, ...
'KFold', 5, ...  %5-fold cross-validation
'ClassNames', {'safe','overturned'});
```



(a) Objective function model



(b) Min. objective vs. number of function evaluations

Figure 4.1: Binary SVM - Bayesian hyper-parameter optimization

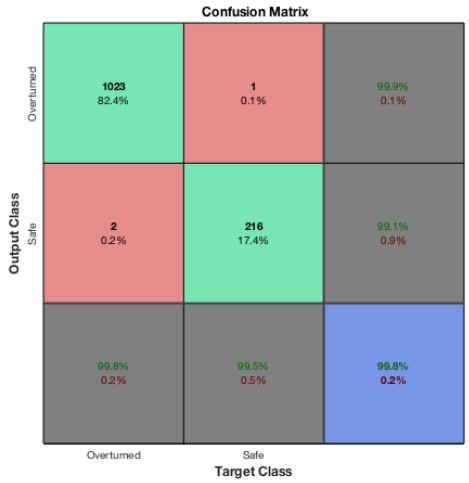


Figure 4.2: Binary SVM - confusion matrix

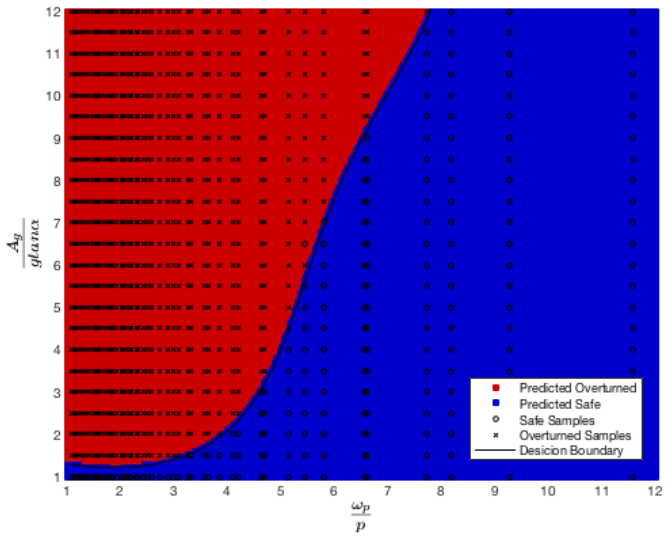


Figure 4.3: Binary SVM - decision surface

Multiclass Classification

For multi-class classification using SVM, the problem is decomposed into several binary classification problems. Thus, a *templateSVM* binary classifier is defined and used in MATLAB's *fitcecoc* function. This function trains and cross-validates a multi-class classification model. The coding method (one-vs-all or one-vs-one) have to be defined.

First of all, an initial multi-class model is constructed, which uses a binary SVM with Gaussian kernel. The optimum hyper-parameters of the binary SVM and the optimum coding method (one-vs-one or one-vs-all) are defined using Bayesian optimization. The objective function is the 5-fold cross-validation loss.

The optimization process is presented in Fig. 4.4. The optimum values for C and σ^2 are 940 and 4.69 respectively. While the best coding method for this data-set is one-vs-one, which means that three binary classifiers will be used to compare each class to another class. The final model is built using these optimzation results. The generalization error of the model is defined with 5-fold cross-validation. The confusion matrix and the decision plane are presented in Fig. 4.5,Fig. 4.6.

Listing 4.2: Three-class classification with support vector machines, training and cross-validation

```
%THREE-CLASS CLASSIFICATION SVM TRAINING
template = templateSVM(... %binary learner
'KernelFunction', 'gaussian', ...
'PolynomialOrder', [], ...
```

```
'KernelScale', 4.68, ...
'BoxConstraint', 940, ...
'Standardize', true, ...
classificationSVM = fitcecoc(...
predictors, ... %input
response, ...   %output
'Coding', 'onevsone',... %onevsone or onvsall
'Learners', template,...
'KFold', 5, ...  %5-fold cross-validation
'ClassNames', {'safe','overturned','overturned-impact'});
```

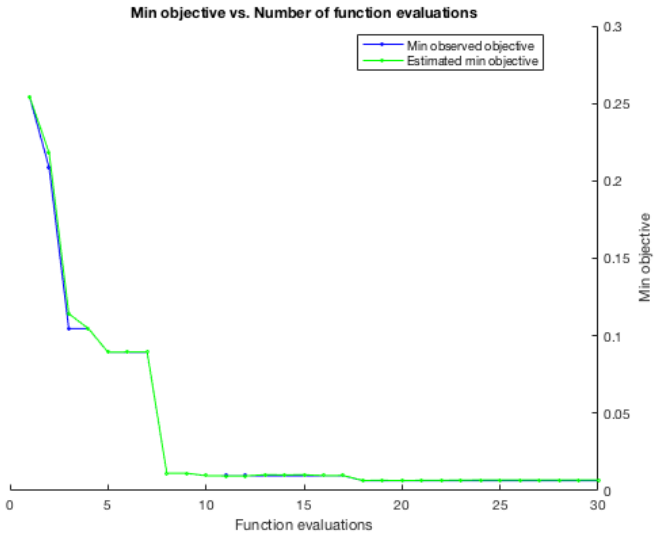


Figure 4.4: Three-class SVM - Bayesian Hyperparameter Optimization

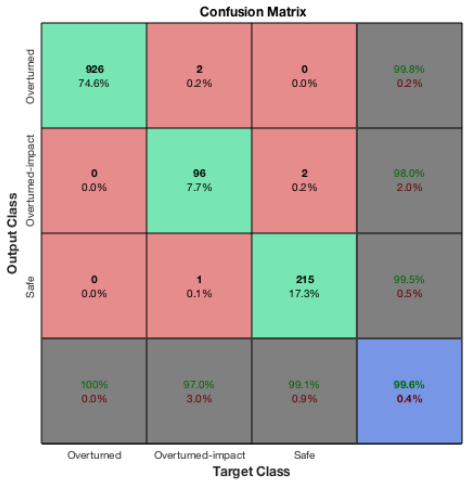


Figure 4.5: Three-class SVM - Confusion matrix

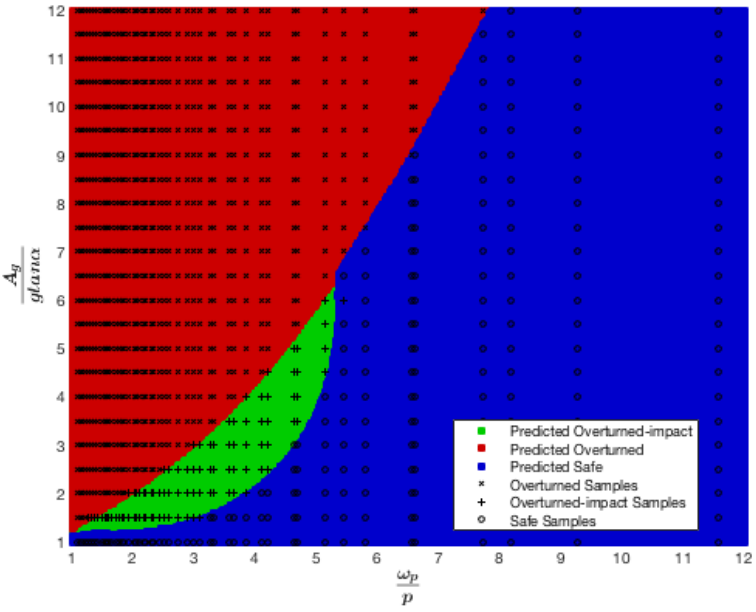


Figure 4.6: Three-class SVM - Decision surface

4.2.2 k-Nearest Neighbours

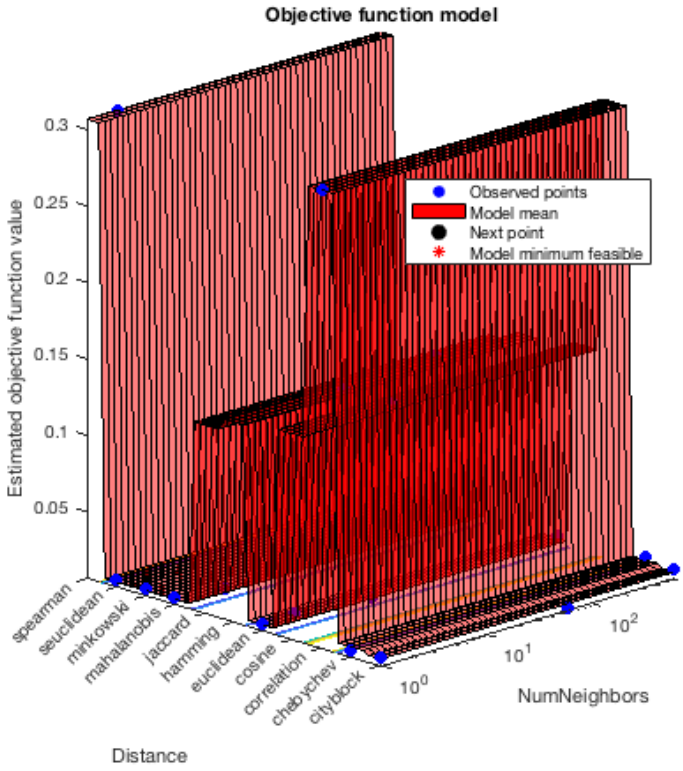
MATLAB's *fitcknn* is used to train k-nearest neighbours classification models and the classification error is calculated with 5-fold cross-validation. The eligible model hyper-parameters for optimization in k-NN is the distance metric and the number of neighbours. It is worth noticing that k-NN algorithm is capable to do multi-class classification without transforming the problem to binary classification problems.

Binary Classification

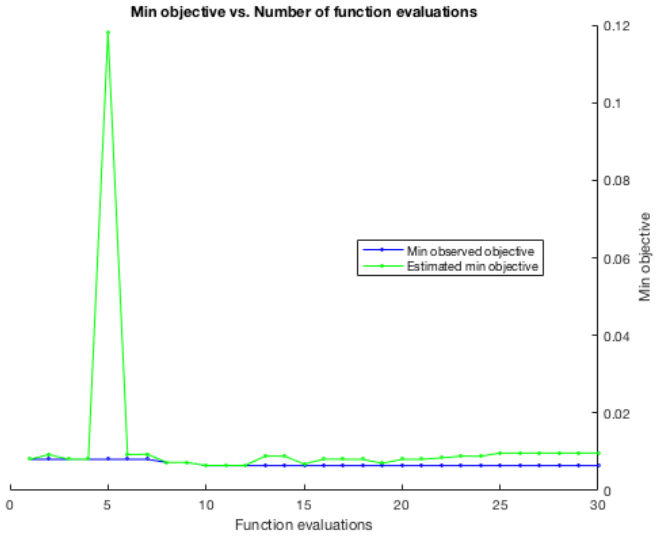
The hyper-parameter optimization results for the binary k-NN model are 11 nearest neighbours and Euclidean distance metric. The hyper-parameter optimization process is shown in Fig. 4.7. The confusion matrix and the decision surface of the model are presented in Fig. 4.8 and Fig. 4.9.

Listing 4.3: Binary classification with k-nearest neighbors, training and cross-validation

```
%BINARY KNN TRAINING
classificationKNN = fitcknn(...
predictors, ... %input
response, ...   %output
'Distance', 'Euclidean', ...
'NumNeighbors', 11, ...
'Standardize', true, ...
'KFold', 5, ...  %5-fold cross-validation
'ClassNames', {'safe','overturned'});
```



(a) Objective function model



(b) Min. objective vs. number of function evaluations

Figure 4.7: Binary k-NN - Bayesian hyper-parameter optimization

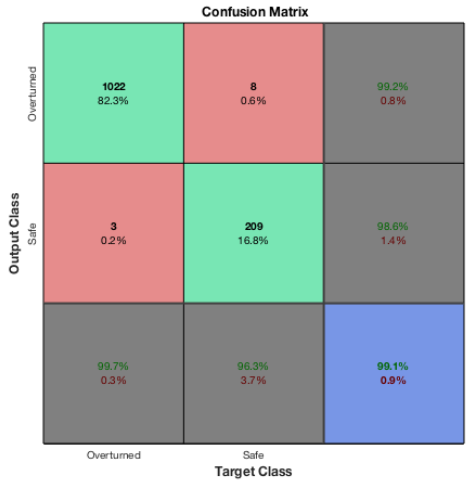


Figure 4.8: Binary k-NN - confusion matrix

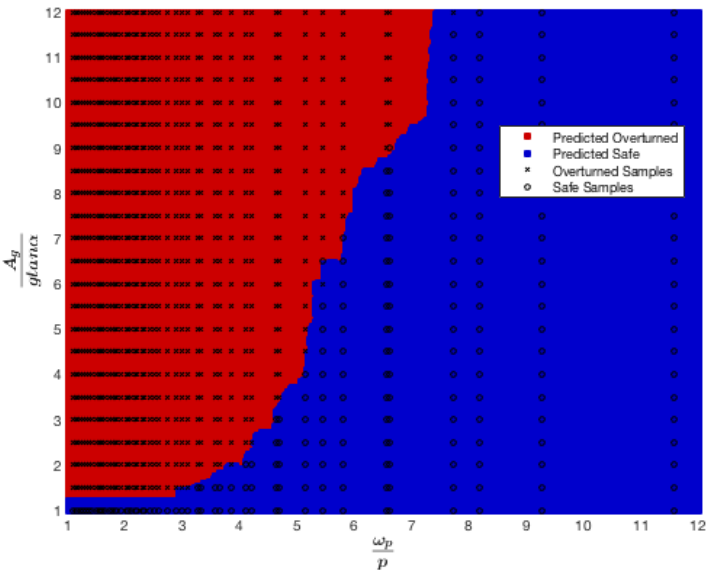


Figure 4.9: Binary k-NN - decision surface

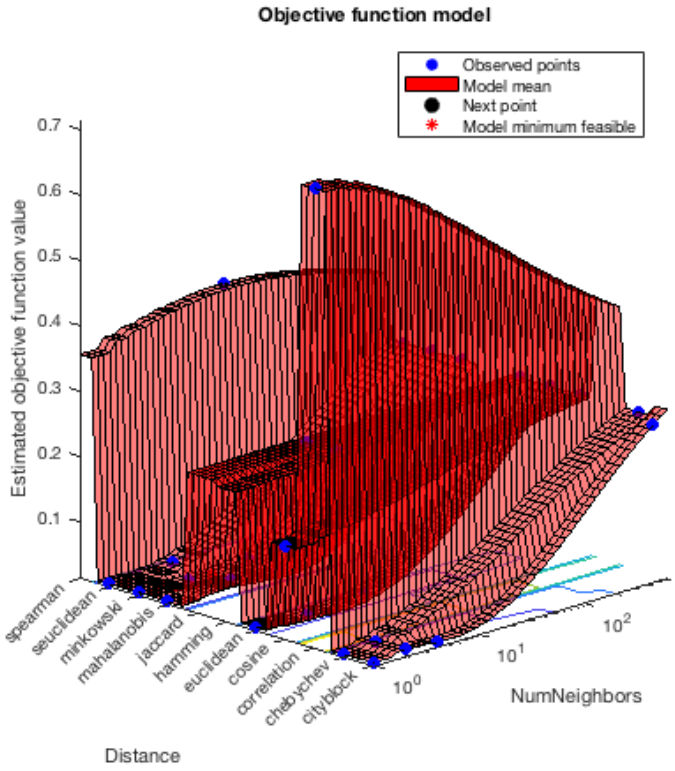
Three-class Classification

The hyper-parameter optimization results for the three-class k-NN model are 5 nearest neighbours and Euclidean distance metric. The hyper-parameter optimization process is shown in Fig. 4.10. The confusion matrix and the decision surface of the model are presented in Fig. 4.11 and Fig. 4.12.

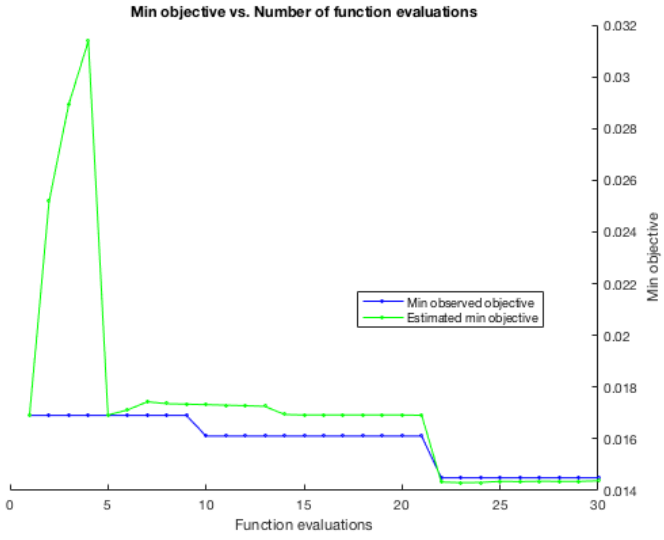
Listing 4.4: Three-class classification with k-nearest neighbors, training and cross-validation

```
%THREE-CLASS KNN TRAINING
classificationKNN = fitcknn(...
predictors, ... %input
response, ...   %output
'Distance', 'Euclidean', ...
'NumNeighbors', 5, ...
'Standardize', true, ...
'KFold', 5, ...  %5-fold cross-validation
'ClassNames', {'safe','overturned','overturned-impact'});
```

The number of neighbours in case of three-class classification is smaller than the binary case. In this way, when the algorithm tries to predict the class of a sample from its k nearest neighbours, avoids including samples from other categories. Although, as can be seen from the decision surface in Fig. 4.12, the model misclassified some points. As mentioned in chapter 2, the k-NN method tries to predict the class of new data points based on the nearest neighbours. Hence, this method is not stable particularly in class borders.



(a) Objective function model



(b) Min. objective vs. Number of function evaluations

Figure 4.10: Three-class k-NN - Bayesian hyper-parameter optimization

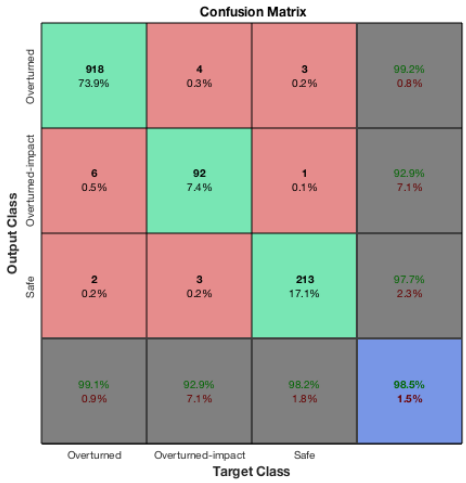


Figure 4.11: Three-class k-NN - Confusion matrix

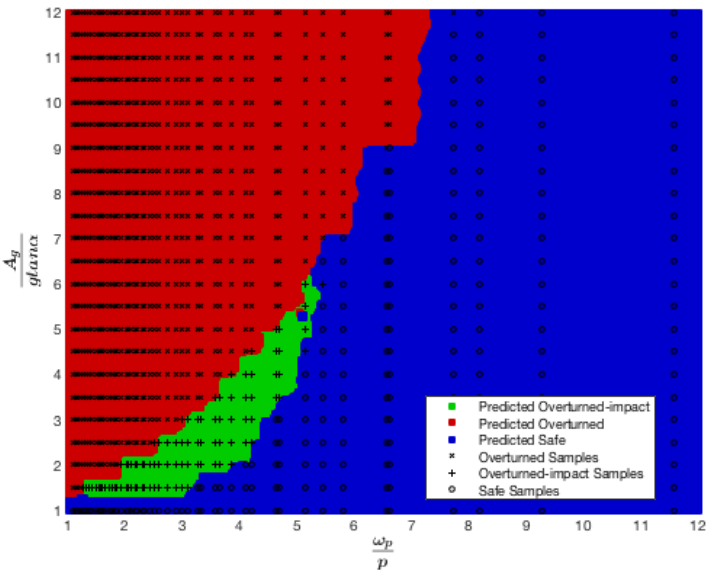


Figure 4.12: Three-class k-NN - Decision surface

4.2.3 Random Forest

In the same manner with the previous models, in random forest case, an initial model is built to define the optimum model hyper-parameters and the final model is built with the optimization results. The model is trained using MATLAB's *fitcensemble*, defining *bagging* method while the classification loss is calculated with 5-fold cross-validation. A random forest consist of several decision trees. Before building the ensemble, the week learner (tree) have to be defined as template object. The hyper-parameters which are optimized in this case are the minimum leaf size and the maximum number of splits of each tree and the number of learning cycles of the ensemble. At every learning cycle, one weak learner (tree) is trained for every defined template object. Consequently, the software trains number of learning cycles∗learners. In our case, only one weak learner is defined. Hence, the number of learning cycles represent the number of trees in the ensemble. It is worth noticing that random forest algorithm is capable to do multi-class classification without transforming the problem to several binary classification problems.

Binary Classification

The hyper-parameter optimization results for the binary random forest are 54 maximum number of splits and 3 minimum leafs at each weak learner and 54 number of trees in the ensemble. The hyper-parameter optimization process is shown in Fig. 4.13. The confusion matrix and the decision surface of the model are presented in Fig.s 4.14, 4.15.

Listing 4.5: Binary classification with random forest, training and cross-validation

```
%BINARY RF TRAINING
template = templateTree(...
'MaxNumSplits', 54,...
'MinLeafSize', 2);
classificationEnsemble = fitcensemble(...
predictors, ... %input
response, ...   %output
'Method', 'Bag', ...
'NumLearningCycles', 10, ...
'Learners', template, ...
'KFold', 5, ...  %5-fold cross-validation
'ClassNames', {'safe','overturned'});
```

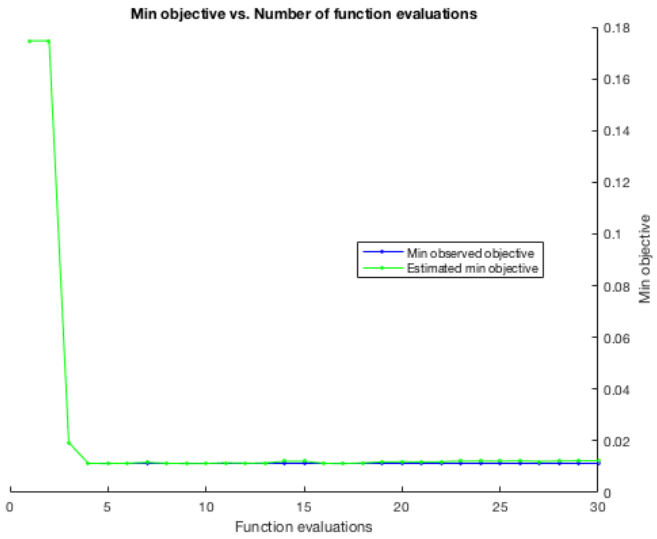


Figure 4.13: Binary random forest-Bayesian hyper-parameter optimization

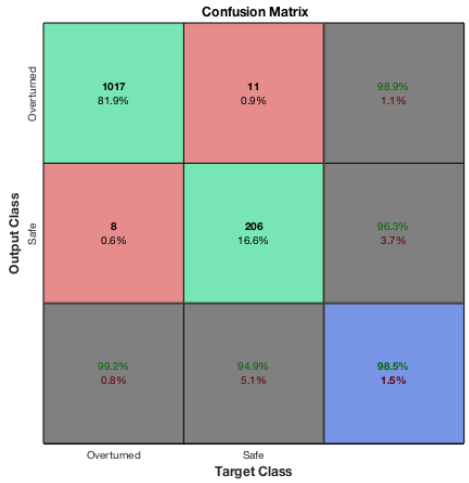


Figure 4.14: Binary RF - confusion matrix

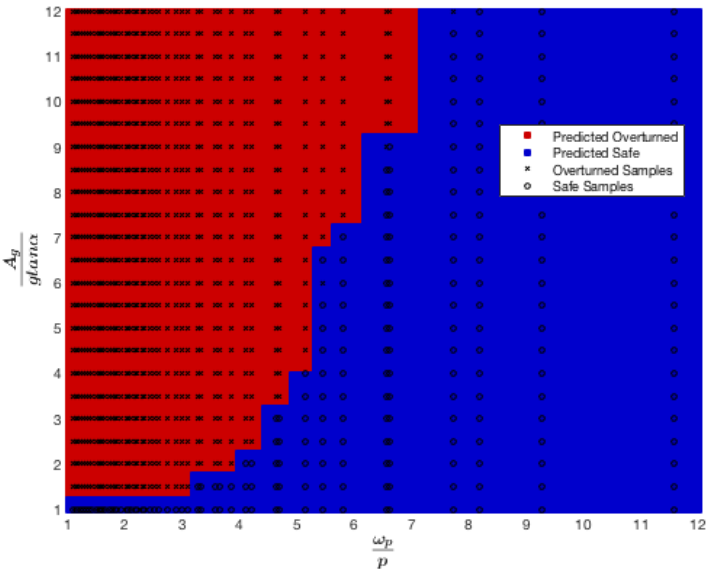


Figure 4.15: Binary RF - decision surface

Three-class Classification

The hyper-parameter optimization results for the three-class random forest are 1238 maximum number of splits and 1 minimum leaf at each weak learner and 187 number of trees in the ensemble. The hyper-parameter optimization process is shown in Fig. 4.16. The confusion matrix and the decision surface of the model are presented in Fig. 4.17 and Fig. 4.18.

The three-class classification problem is more complex from the binary classification problem. Thus, in this case the depth of a single decision tree in the random forest is bigger. The depth of a single decision tree is controlled by the number of splits and the number of leaf nodes. On the other hand, for both cases 10 decision trees are enough to form a random forest classifier.

Listing 4.6: Three-class classification with random forest, training and cross-validation

```
%BINARY RF TRAINING
template = templateTree(...
'MaxNumSplits', 75,...
'MinLeafSize', 3);
classificationEnsemble = fitcensemble(...
predictors, ... %input
response, ...   %output
'Method', 'Bag', ...
'NumLearningCycles', 187, ...
```

```
10 'Learners', template, ...
11 'KFold', 5, ...  %5-fold cross-validation
12 'ClassNames', {'safe','overturned','overturned-impact'});
```

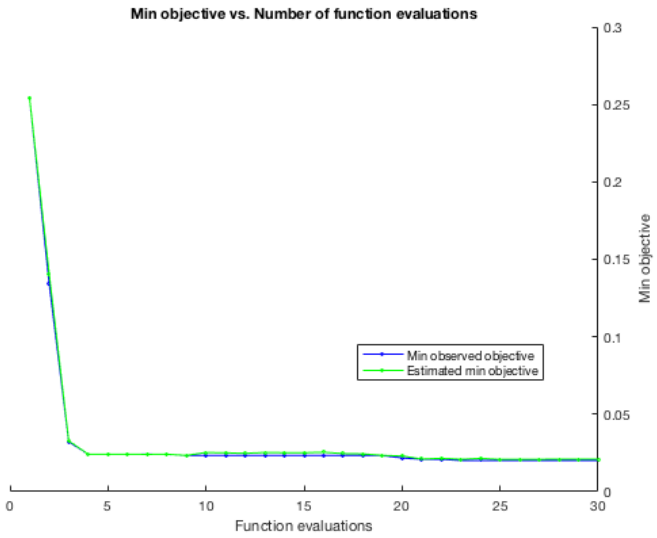


Figure 4.16: Three-class random forest-Bayesian hyper-parameter optimization

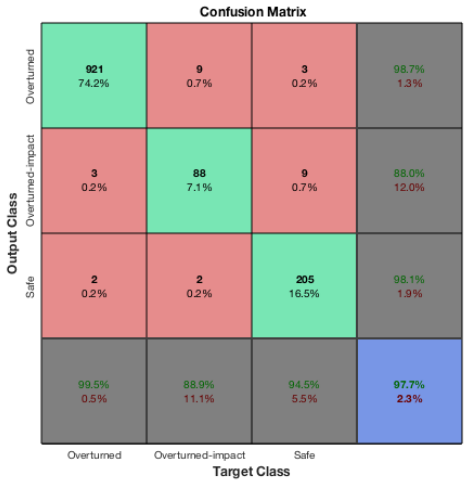


Figure 4.17: Three-class RF - Confusion matrix

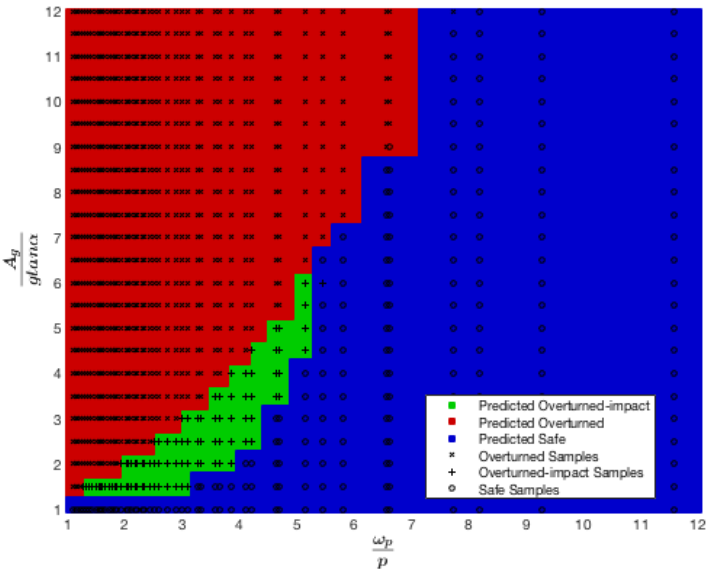


Figure 4.18: Binary RF - Decision surface

4.2.4 Comparison of Classification Models

The results of the machine learning models for the two classification tasks on the same training data are presented analytically in the previous sections. The results of the models are grouped together in the next figures, in order to make easier the comparison of their performance. Specifically, Fig. 4.19 and Fig. 4.20 display the confusion matrices of support vector machien, k-nearest neighbours, random forest and their decision surfaces for the binary classification problem. Fig. 4.21 and Fig. 4.22 display the confusion matrices and the decision surfaces for the three-class classification problem. The accuracy of the models is shown in tables 4.4 and 4.5.

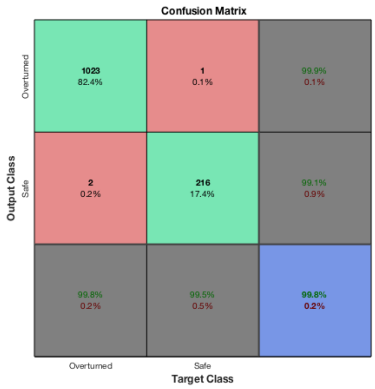
Support vector machines have the highest overall accuracy for both binary and three-class classification problems for the examined data-set. Besides the accuracy, they are preferable because of the concrete decision boundary they form compared to random forest and k-nearest neighbours decision boundaries. As mentioned above, k-NN is unstable at class borders because of its instance-based predicting process. On the other hand, random forests have the less smooth decision boundary. The reason for this is that in a similar way with k-NN, the boundary of a random forest is formed from the average of all the predictions from different trees.

Table 4.4: Binary classification results

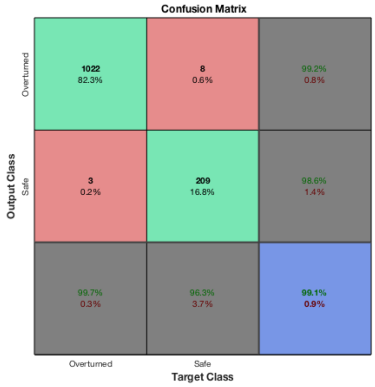
Model	**Accuracy(%)**
Support Vector Machine	99.6
k-Nearest Neighbours	99.1
Random Forest	98.5

Table 4.5: Three-class classification results

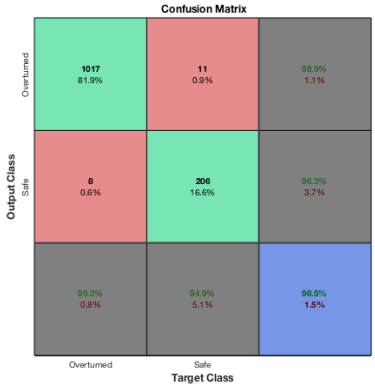
Model	**Accuracy(%)**
Support Vector Machine	99.6
k-Nearest Neighbours	98.5
Random Forest	97,7



(a) SVM

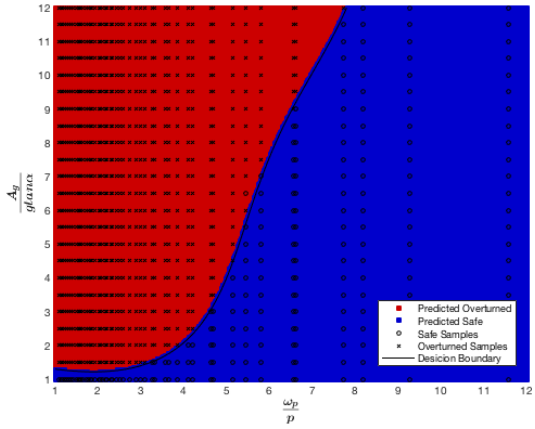


(b) k-NN

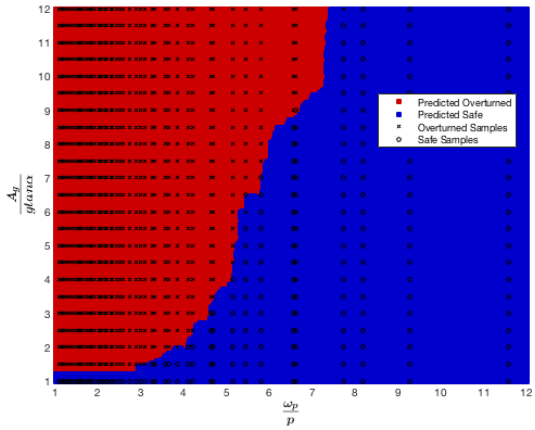


(c) Random Forest

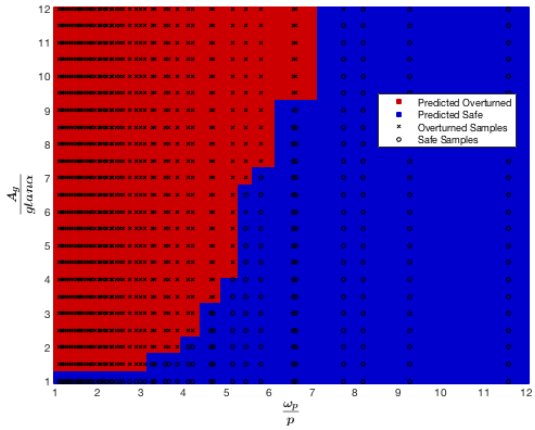
Figure 4.19: Binary classification - Confusion Matrices



(a) SVM

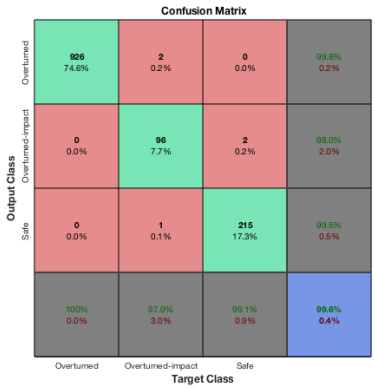


(b) k-NN

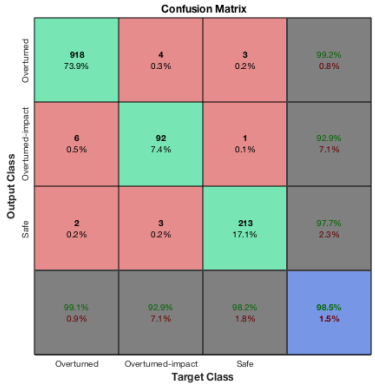


(c) Random Forest

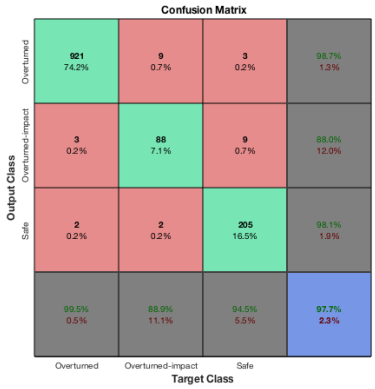
Figure 4.20: Binary Classification - Decision Surfaces



(a) SVM

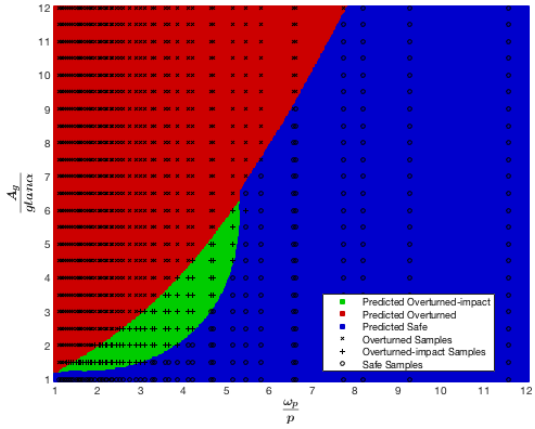


(b) k-NN

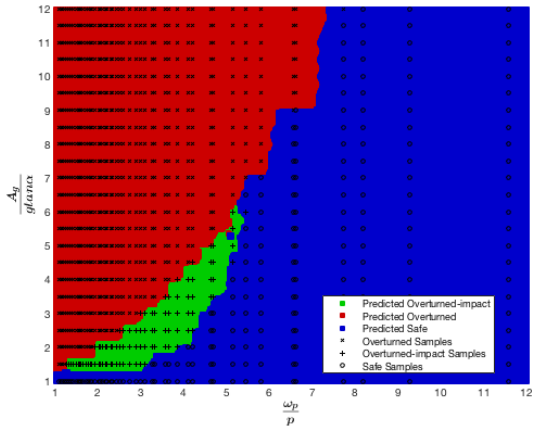


(c) Random Forest

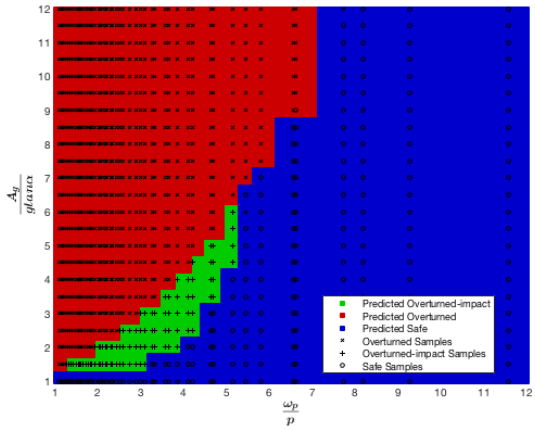
Figure 4.21: Three-class Classification - Confusion Matrices



(a) SVM



(b) k-NN



(c) Random Forest

Figure 4.22: Three-class classification - Decision Surfaces

4.3 Optimization of K-Nearest Neighbours and Random Forest Decision Surfaces

As explained in the previous section, k-nearest neighbours and random forest algorithms form their decision surfaces based on the instances in the given training data-set, they are unstable at class borders and they tend to overfit. In order to get smoother decision surfaces, more training data points are added to the initial training set.

First of all, the class border points are defined. Then, totally 2100 input feature pairs are generated using normal distribution with mean the points of the border curve and co-variance 0.2, Fig.4.23. The input points are solved in the same manner with the initial training data-set. The new points are added to the initial data-set. The new training data-set has totally 3342 input-output pairs.

Random forest and k-nearest neighbors algorithms for three-class classification are trained with the new training data-set. The results are shown in Fig.4.24 and Fig.4.25. The overall accuracy of the models remain the same. However, their decisions in regions next to class borders are improved and the decision boundaries are smoother. Support vector machines are still preferable considering that they achieve good accuracy and smoother boundaries without adding more training data.

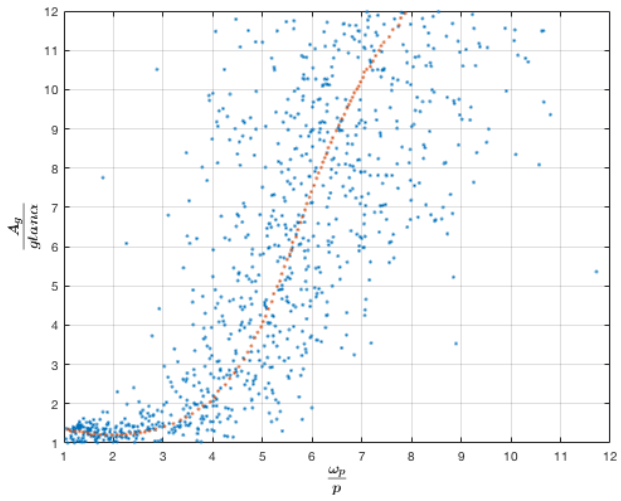


Figure 4.23: Generated training data points close to class borders

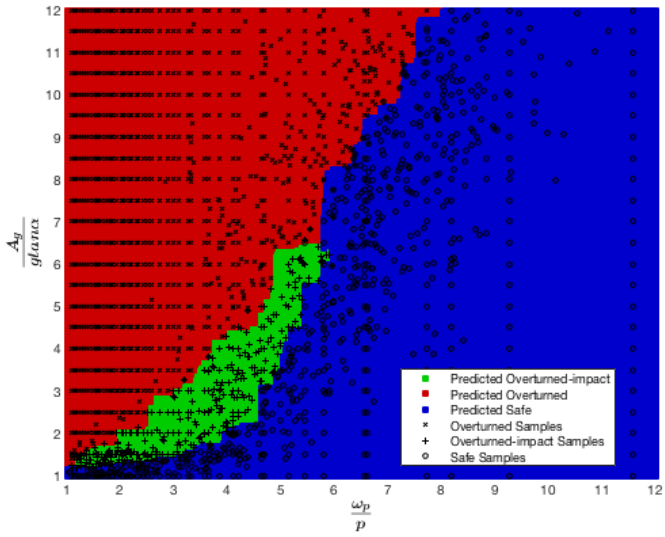


Figure 4.24: Three-class random forest - decision surface after adding more data points

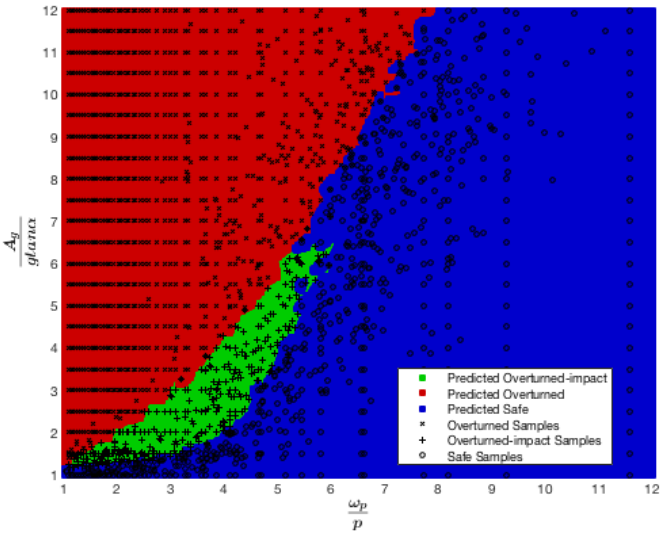


Figure 4.25: Three-class k-nearest neighbors - training results after adding more data points

4.4 Multiclass Classification with Support Vector Machines

As analyzed in the previous sections, support vector machines have the best performance for classification tasks for the examined data-set without needing to add more training data points. They form the smoothest decision boundaries without tending to overfit. For this reason, they are used for more complex classification tasks with the initial training data-set (1242 samples). Specifically, four class classification and five class classification models are built and they are presented below.

The training data-set is split into four categories as presented below:

- Overturning block : $\frac{\theta}{\alpha} \geq 1 \rightarrow$ "*overturned*"
- High rotation ratio : $0.5 < \frac{\theta}{\alpha} < 1 \rightarrow$ "*high rotation*"
- Medium rotation ratio : $0.2 < \frac{\theta}{\alpha} < 0.5 \rightarrow$ "*medium rotation*"
- Low rotation ratio : $\frac{\theta}{\alpha} < 0.2 \rightarrow$ "*low rotation*"

In five-class classification, the overturned blocks are also classified as overturning with ('*overturned*) and without ($o - impacts$) impacts.

The decision boundaries and the confusion matrices of the four-class and five-class classification models are presented in Fig. 4.26, Fig. 4.27 and Fig. 4.28, Fig. 4.29, respectively. As it was expected, the models have a very high overall accuracy, 98.6% and 98.0% and they form very smooth decision boundaries.

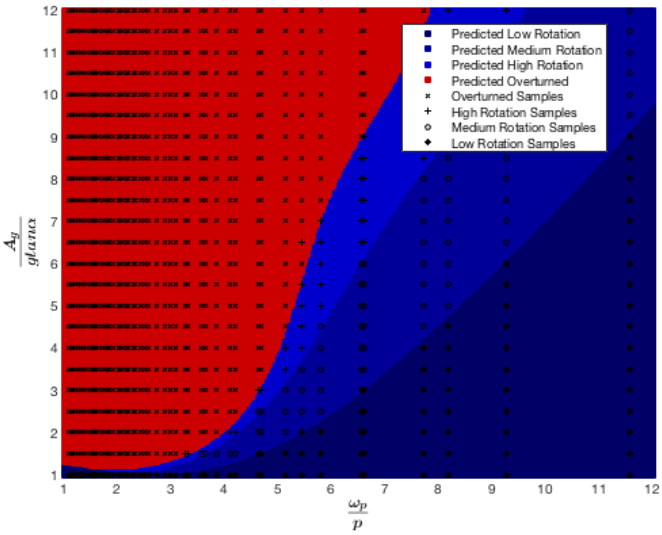


Figure 4.26: Four-class SVM - decision surface

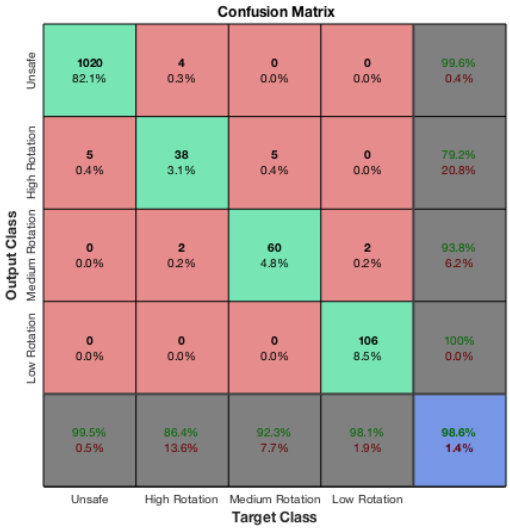


Figure 4.27: Four-class SVM - confusion matrix

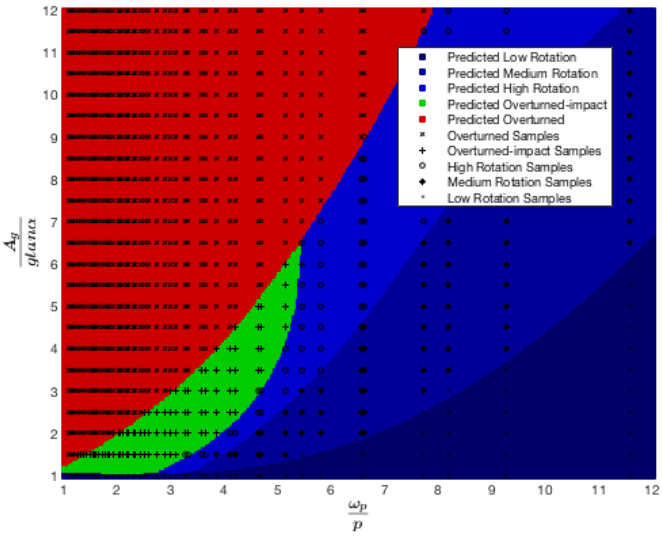


Figure 4.28: Five-class SVM - decision surface

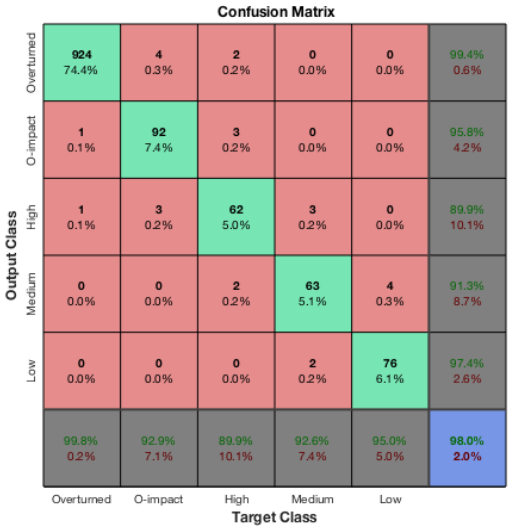


Figure 4.29: Five-class SVM - confusion matrix

4.5 Regression Models

4.5.1 Gaussian Process Regression

MATLAB's *fitrgp* function trains and cross-validates a Gaussian process regression model. Fitting a GPR model involves estimating the kernel function parameterized in terms of kernel parameters σ_f and σ_l, the noise variance σ^2 (Sigma) and coefficient vector of fixed basis functions from the training data. Bayesian optimization is used to estimate the optimum kernel and basis functions and the noise variance σ^2 for the examined data-set, as shown in Fig. 4.30. The objective function of the optimization is *log(1 + cross-validation loss)*, as it is in any regression model.

The root mean square error, RMSE of the model is 0.0089. In order to observe the models predictions for different values of the output $\frac{\theta}{\alpha}$, the predicted vs. actual plot and the residuals plot are shown in Fig. 4.31 and Fig. 4.32.

Listing 4.7: Gaussian process regression, training and cross-validation

```
%Gaussian Process Regression Training
regressionGP = fitrgp(...
predictors, ...
response, ...
'BasisFunction', 'none', ...
'KernelFunction', 'matern52', ...
'Standardize', false, ...
'Sigma', 0.000133305, ...
'KFold, 5);
```

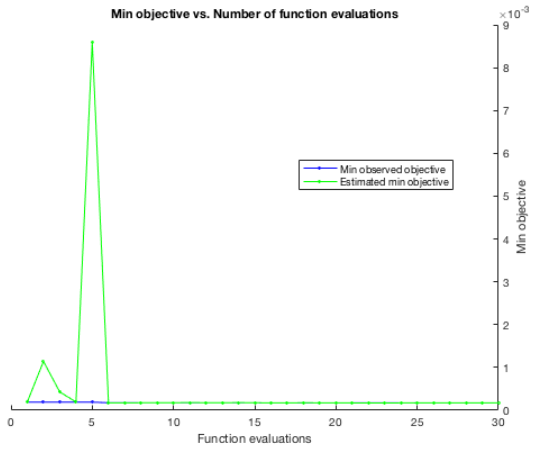


Figure 4.30: Gaussian process regression - Bayesian hyper-parameter optimization

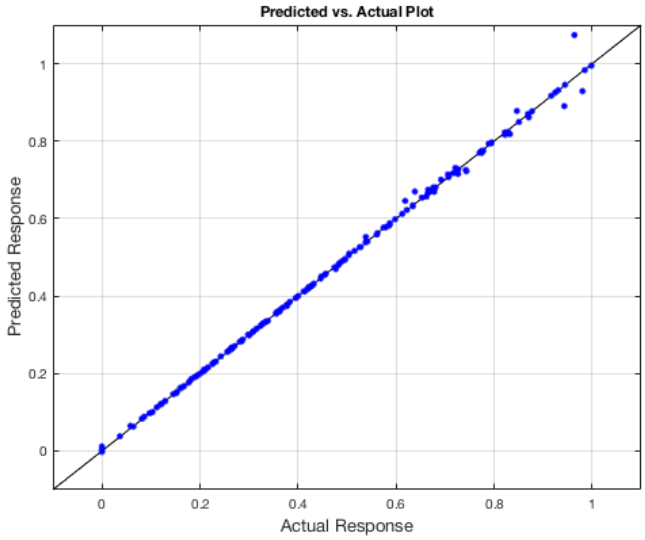


Figure 4.31: GPR - Predicted vs. Actual Plot

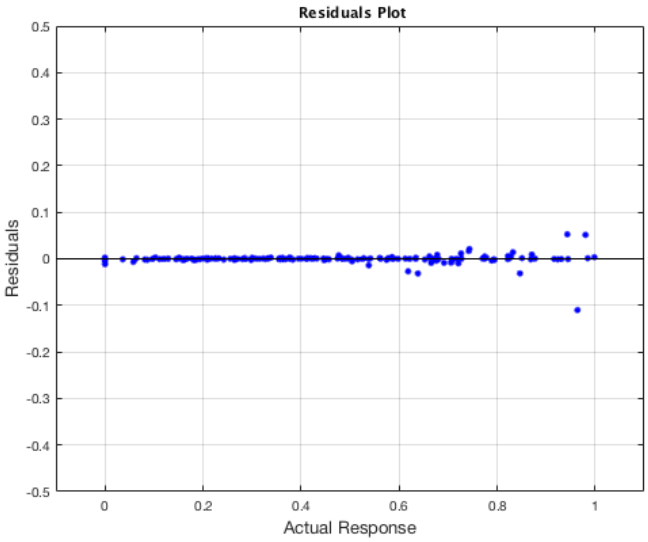


Figure 4.32: GPR - Residual vs. Actual Plot

4.5.2 Random Forest

The hyper-parameters of the random forest regression which are optimized are the same with the classification case. The Bayesian optimization process is displayed in Fig 4.33. The RMSE of the model is 0.09.

Listing 4.8: Random forest regression, training and cross-validation

```
%Random Forest Regression Training
template = templateTree(...
'MinLeafSize', 1, ...
'MaxNumSplits', 552);
regressionEnsemble = fitrensemble(...
predictors, ...
response, ...
'Method', 'Bag', ...
'NumLearningCycles', 497, ...
'Learners', template, ...
'KFold' , 5);
```

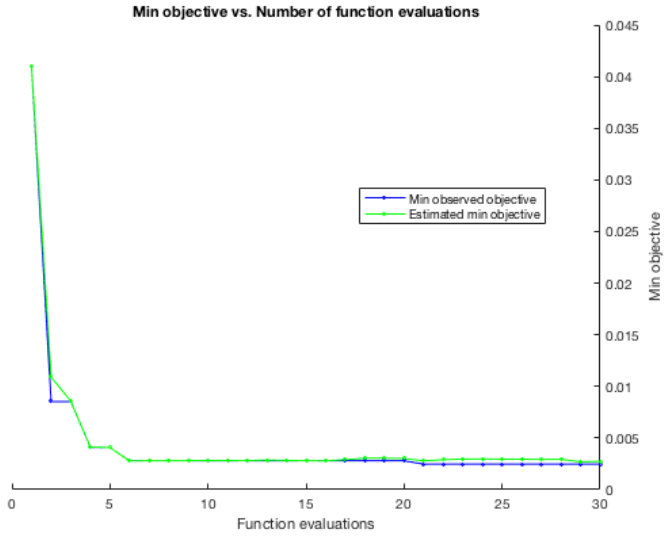


Figure 4.33: Random Forest Regression - Bayesian Hyperparameter Optimization

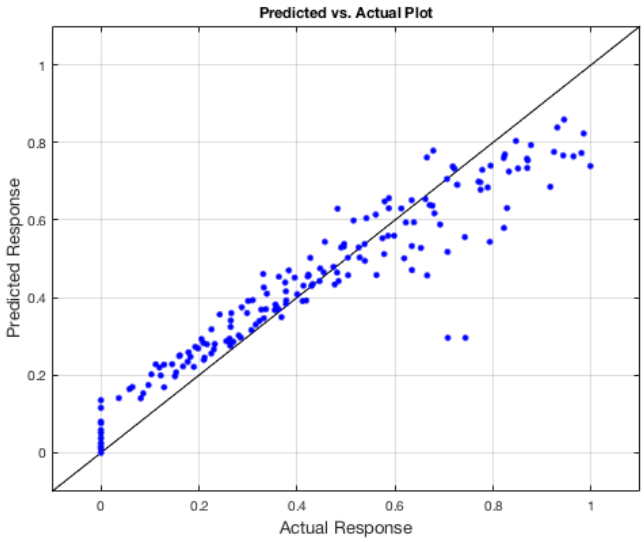


Figure 4.34: Random Forest - Predicted vs. Actual Plot

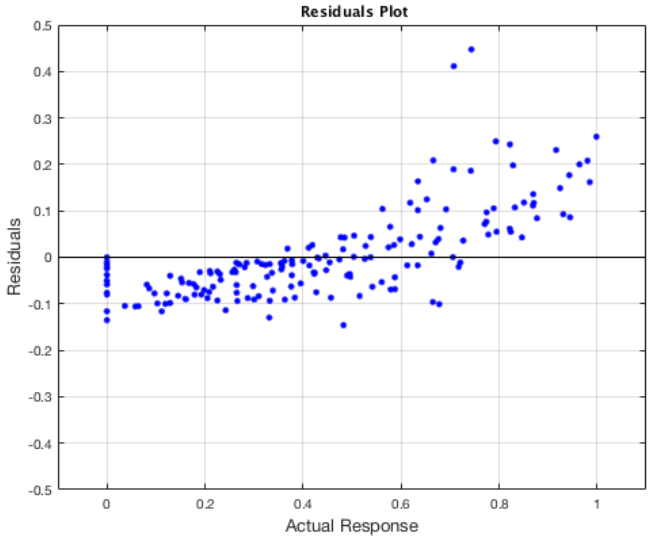


Figure 4.35: Random Forest - Residual vs. Actual Plot

4.5.3 Support Vector Regression

Support vector regression hyper-parameter which are optimized are the same with the classification case plus the margin of tolerance ϵ. The Bayesian optimization process is displayed in Fig 4.36. The RMSE of the model is 0.057.

Listing 4.9: Support Vector Regression, Training and Cross-validation

```
%Support Vector Regression Training
responseScale = iqr(response);
if ¬isfinite(responseScale) || responseScale ==0.0
responseScale = 1.0 ;
```

```
5  end
6  boxConstraint = responseScale/1.349;
7  epsilon = responseScale/13.49;
8  regressionSVM = fitrsvm (...
9  predictors, ...
10 response, ...
11 'KernelFunction' , 'gaussian', ...
12 'PolynomialOrder, [], ...
13 'KernelScale', 0.35, ...
14 'BoxConstraint', boxConstraint, ...
15 'Epsilon', epsilon, ...
16 'KFold', 5, ...
17 'Standardize', true);
```

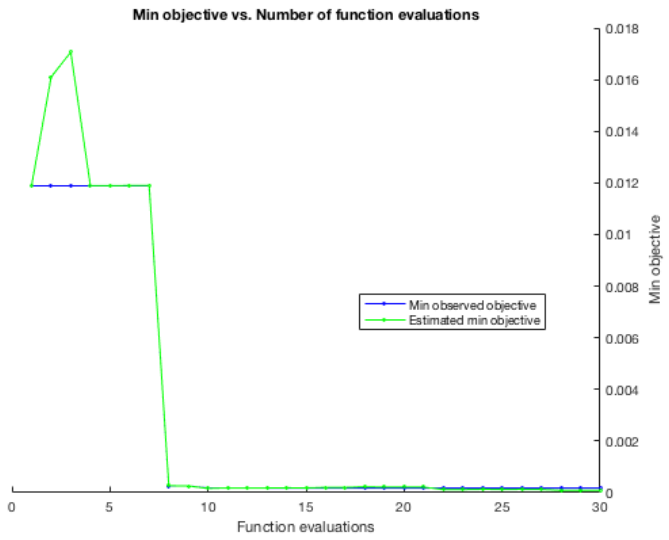


Figure 4.36: Support Vector Regression - Bayesian Hyperparameter Optimization

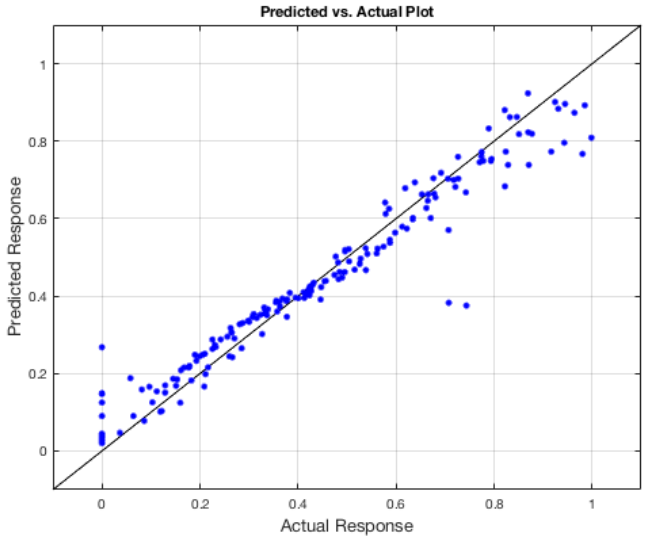


Figure 4.37: SVR - Predicted vs. Actual Plot

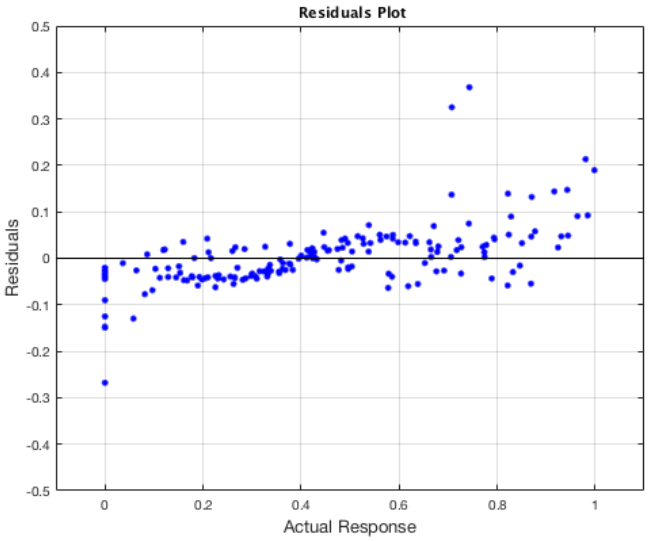
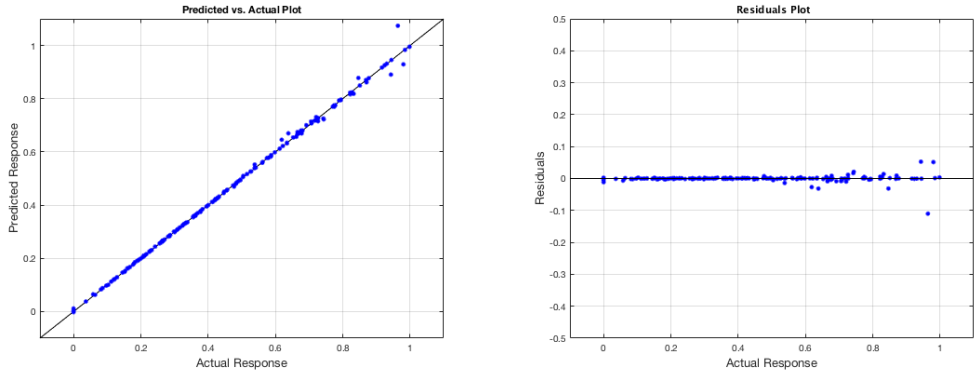


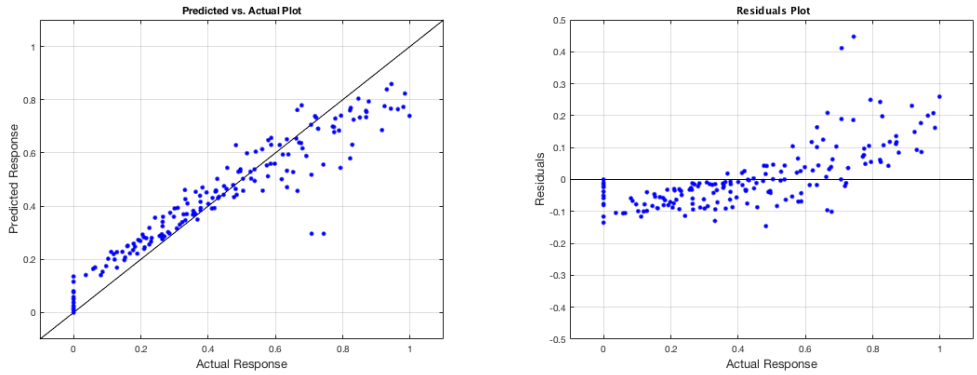
Figure 4.38: SVR - Residual vs. Actual Plot

4.5.4 Comparison of Regression Models



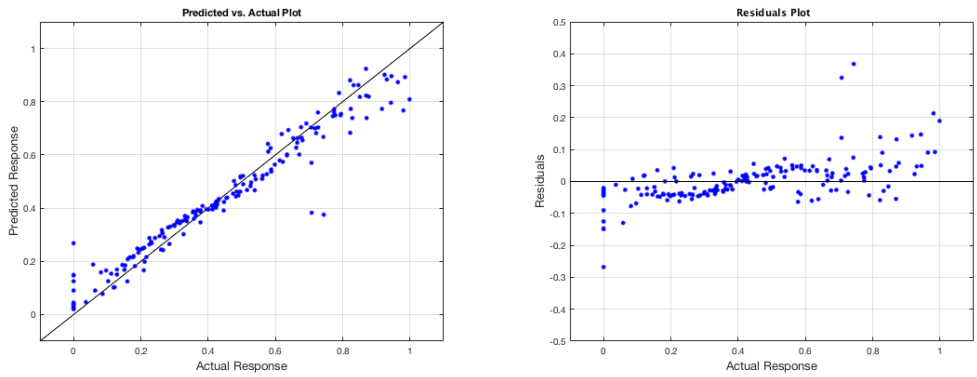
(a) GPR

(b) GPR



(c) RF

(d) RF



(e) SVM

(f) SVM

Figure 4.39: Regression - Performance Plots

The RMSE of each model for the regression problem is presented in Table 4.6 below.

Table 4.6: Regression Results

Model	**RMSE**
GPR	0.0089
RF	0.09
SVM	0.057

As can be seen at the performance plots, the Gaussian process regression model has the best performance. Besides the RMSE which is very low, the performance plots of the model indicate that it perfectly fits the training data and accurately predicts the block's rotation ratio $\frac{\theta}{\alpha}$.

Random forest and support vector regression models have also very satisfying RMSE. However, they tend to overestimate the $\frac{\theta}{\alpha}$ values lower than 0.5. They also underestimate $\frac{\theta}{\alpha}$ values greater that 0.5. Underestimating block's rotation within the range of high values is considered very problematic. Thus, the Gaussian process regression model is preferred to make predictions about the response of new block samples under the same conditions with the considered data-set. Furthermore, the application of it to rigid block's subjected to near-fault ground motion will be discussed in the next chapter.

Chapter 5: Rocking Blocks Subjected to Near Fault Ground Motion

As presented in Chapter 4, Gaussian process regression has the best performance for regression tasks. In this chapter, this algorithm is used to predict the response of rigid blocks which are subjected to near fault ground motion. The preparation of the data-set is presented in the next section. In this case, the possible input parameters of the training are much more compared to the previous data-set, because of the complexity of the problem. Thus, input feature (predictor) importance estimation is performed using the random forest algorithm.

5.1 Data-set Preparation

First step of the data-set preparation is the extraction of Mavroeidis and Papageorgiou wavelets [22], from near-fault ground motion records, 55 records in this case. The above mentioned researches have proposed a simple analytical model for the representation of near-field strong ground motions. The model adequately describes the nature of the impulsive near-fault ground motions both qualitatively and quantitatively. In addition, it may be used to analytically explain empirical observations that are based on available near-source records.

The input parameters of the model have an unambiguous physical meaning. The proposed analytical model has been calibrated using several near-field ground

motion records. It successfully simulates the entire set of available near-fault displacement, velocity, and (in many cases) acceleration time histories, as well as the corresponding deformation, velocity, and acceleration response spectra.

Four parameters of the velocity pulse, which are mentioned below, are used as input parameters of the model and acceleration, velocity and displacement time histories are generated for each 55 records.

- A_p: Velocity amplitude of the envelope of the signal.
- f_p: Frequency of the amplitude-modulated harmonic (prevailing angular frequency of the signal).
- $T_p = \frac{1}{f_p}$: Pulse duration.
- ν_p : Phase of the amplitude-modulated harmonic, i.e. $\nu = 0$ and $\nu = \frac{\pi}{2}$ define symmetric and antisymmetric signals, respectively.
- γ_p : Oscillatory character (i.e.number of half cycles) of the signal.

Next step after the generation of Mavroeidis and Papageorgiou wavelets for the 55 natural ground motion records, is the assessment of the response of rigid blocks under the excitation of the near fault ground motion. Blocks with different sizes (R=0.5÷2m) and energy dissipation $\eta = 0.85$ are subjected to 55 near fault ground motion records and the block rotation ratio over the slenderness angle ($\tan\alpha = 0.33$) is calculated for totally 510 block samples. As in the previous dataset, the equation of motion of the block is solved using ODE23s solver available in MATLAB.

The considered input features for the regression model are the parameters of the Mavroeidis and Papageorgiou wavelets, block's characteristic frequency p, the excitation frequency $\omega_p = \frac{2\pi}{T_p}$, the amplitude of acceleration A_g and the residual pseudo-spectral velocity, $PSVres$. The output of the model is the block's rotation ratio over the slenderness angle $\frac{\theta}{\alpha}$, for $\frac{\theta}{\alpha} < 1$. The training data-set for the considered regression model is presented in Table 5.1.

Table 5.1: Blocks under near-fault ground motion - training set

Input	**Output**
$\frac{\omega_p}{p}$	
$\frac{A_g}{gtan\alpha}$	
T_p	
A_p	$\frac{\theta}{\alpha}$
f_p	
ν_p	
γ_p	
$PSVres$	
Number of Samples	510

5.2 Regression

The parameters of the trained Gaussian process regression model are presented in Table 5.2. The RMSE of the model is 0.0686. The predicted vs. actual plot and the residuals plot are shown in Fig. 5.1 and Fig. 5.2, respectively.

The general performance of the model is quite satisfying. As it was expected, the residuals in case of blocks under near-fault ground motion are higher compared to blocks under one sine excitation. For values $\frac{\theta}{\alpha} > 0.4$ the model tends to underestimate $\frac{\theta}{\alpha}$. While for $\frac{\theta}{\alpha} < 0.6$, there are some outliers that overestimate the value of $\frac{\theta}{\alpha}$.

Table 5.2: Gaussian Process Regression Parameters for Near Fault Ground Motion

Kernel Function	ardrationalquadratic
Basis Function	none
Sigma	0.00010229
Cross-Validation	5-fold

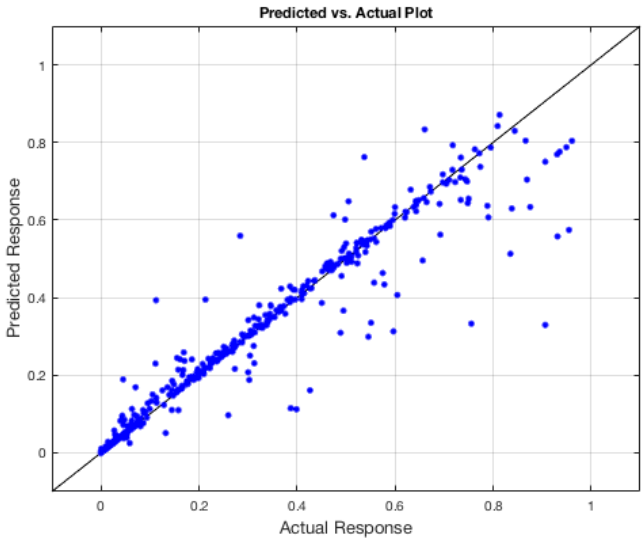


Figure 5.1: GPR - Predicted vs. Actual Plot

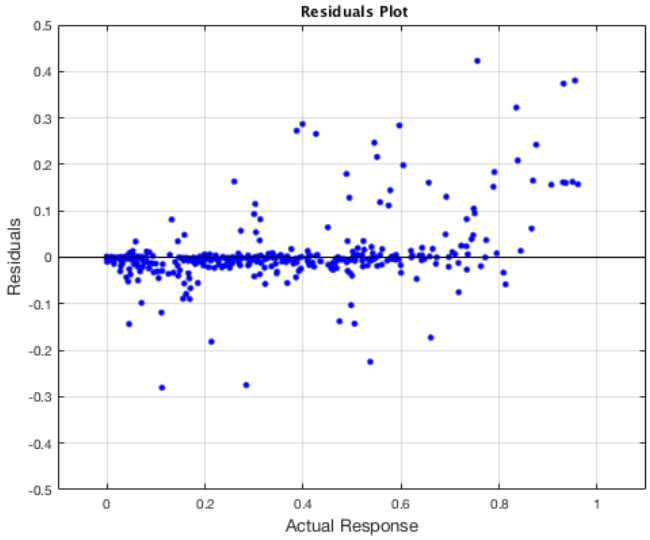


Figure 5.2: GPR - Residuals Plot

5.3 Predictor Importance Estimation

In training data-sets with many input features, another useful machine learning task which can be applied is the estimation of the importance of the input features. There are many methods available to achieve this estimation. In this dissertation, out-of-bag predictor importance estimation by permutation is applied using the random forest algorithm. The related process is discussed analytically in section 2.2.3.5.

The results are presented in Fig. 5.3. As it was expected, the biggest influence in the predictions of the random forest model have the input features $\frac{\omega_p}{p}$ and $\frac{A_p}{gtan\alpha}$. The residuals of the pseudo-spectral velocity has also influence in the predictions

of the model. The less influential input features are the parameters of the wavelet.

Listing 5.1: Predictor Importance Estimation using Random Forest Algorithm

```
1  inputTable = array2table(reg_data_nf, 'VariableNames', ...
       {'omegapp','pgagtana','Tp','Ap','gamma','ni','psvres','theta'});
2  predictorNames ...
       ={'omegapp','pgagtana','Tp','Ap','gamma','ni','psvres'};
3
4  predictors = inputTable(:, predictorNames);
5  response = inputTable.theta;
6  isCategoricalPredictor = [false, false];
7
8  t = templateTree('PredictorSelection','curvature');
9  ens = fitrensemble(predictors,response,'Learners',t, ...
       'Method','bag','NumLearningCycles',500);
10
11 options = statset('UseParallel',true);
12 imp = oobPermutedPredictorImportance(ens,'Options',options);
```

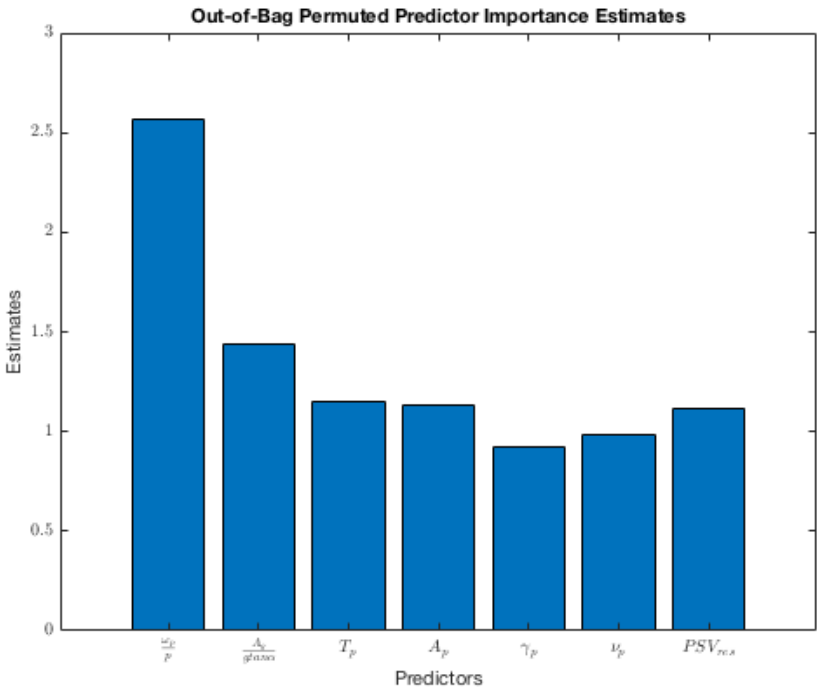


Figure 5.3: Predictor Importance Estimation

Chapter 6: Conclusion Remarks

The purpose of this master's dissertation was to investigate the application of several machine learning algorithms to rocking problems. Specifically, a number of machine learning algorithms were trained appropriately to predict the seismic response of rocking blocks subjected to one-sine pulse base excitation. In order to prepare the required training data, the equation of motion of various rigid blocks was solved using standard ODE solvers available in MATLAB for varying geometric properties and excitation characteristics. Then, machine learning algorithms were trained with this training data to make accurate predictions about the rocking response of new blocks and their performance were compared. The study was extended to blocks subjected to near-fault ground motion using the best performing algorithms.

Initially, machine learning algorithms were used for classification and regression problems for rigid blocks subjected to one-sine base excitation. Two types of classification tasks were examined, binary and three-class classification. In binary classification, the blocks were classified as safe and overturned. In three-class classification, the blocks are classified as safe, overturned and overturned with impact. The algorithms which are used for classification were support vector machines, k-nearest neighbors and random forest. As presented in chapter 4, the best performing algorithm was the support vector machines. Support vector machines had

the highest overall accuracy for both binary and three-class classification problems for the examined data-set. Besides the accuracy, they are preferable because of the smooth decision boundary they form compared to random forest and k-nearest neighbours decision boundaries. In order to get smoother decision boundaries with k-nearest neighbors and random forest algorithms more training data points were added to the initial training set close to the class borders. After this process, their decision boundaries get smoother without increasing the overall accuracy of the models. As the best performing algorithm, support vector machines were used for four-class and five-class classification using the same training data-set.

Gaussian process regression, support vector regression and random forest algorithms were used for predicting the block's rotation for the same data-set. As can be seen at the performance plots presented in chapter 4, the Gaussian process regression model had the best performance. Besides the RMSE which was very low, the performance plots of the model indicate that it perfectly fits the training data and accurately predicts the block's rotation ratio $\frac{\theta}{\alpha}$. Random forest and support vector regression models have also very satisfying RMSE. However, they tend to overestimate the $\frac{\theta}{\alpha}$ values lower than 0.5. They also underestimate $\frac{\theta}{\alpha}$ values greater that 0.5. Underestimating block's rotation within the range of high values is considered very problematic. Thus, the Gaussian process regression model is preferred to make predictions about the response of new block samples.

The study was extended to blocks under near-fault ground motion records. The best performing algorithm for regression is applied for this case. Gaussian process regression algorithm had a quite satisfying performance. However, the

model needs more investigation and optimization to achieve better results. Taking into account that the examined data-set had many input features, their importance estimation is accomplished using random forest algorithm. The biggest influence in the predictions of the model had the input features $\frac{\omega_p}{p}$ and $\frac{A_p}{gtan\alpha}$. The case of rigid

As a summary, support vector machines are considered as the best performing algorithms for classification applied to rocking blocks under one-sine excitation. For regression tasks, Gaussian process regression algorithm was the best performing algorithm. The application of them to blocks under near-fault ground motion had also adequate results but could be optimized further. Also, random forest algorithm would be a suitable choice to estimate the importance of the input features in data-sets with many input features. The application of classification algorithms, as well as the further investigation of regression task for blocks under real earthquake records could be interesting ideas for future research.

Bibliography

[1] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] S. J. Russell and P. Norvig. *Artificial intelligence: A modern Approach, Third Edition*. Prentice Hall, 2010.

[3] N. Christianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge, UK: Cambridge University Press, 2000.

[4] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning, second edition*. New York: Springer, 2008.

[5] R. Rifkin and A. Klautau. Parallel networks that learn to pronounce english text. *Journal of Machine Learning Research*, 2004.

[6] T. Hastie and R. Tibshirani. Classification by pairwise coupling. *Advances in Neural Information Processing Systems*, 1998.

[7] N. S. Altman. *An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression*. The American Statistician, 1992.

[8] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth & Brooks/Cole Advanced Books & Software, 1984.

[9] Hemlata C. ID3 Modification and Implementation in Data Mining. *International Journal of Computer Applications*, 80(7), 2013.

[10] L. Breiman. Bagging Predictors. *Machine Learning*, 1996.

[11] T. K. Ho. A Data Complexity Analysis of Comparative Advantages of Decision Forest Constructors. *Pattern Analysis and Applications*, 2002.

[12] W.Y. Loh and Y.S Shih. Split Selection Methods for Classification Trees. *Statistica Sinica*, 1997.

[13] Platypus Innovation. A simple intro to gaussian processes. http://platypusinnovation.blogspot.com/2016/05/a-simple-intro-to-gaussian-processes.html.

[14] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[15] E. C. Sammut and G. I. Webb. Biasvariance decomposition, In Encyclopedia of Machine Learning. *Springer*, 2011.

[16] G.w. Housner. The behavior of inverted pendulum structures during earthquakes. *Bulletin of the Seismological Society in America*, 1963.

[17] C.S. Yim, A. K. Chopra, and J. Penzien. Rocking Response of Rigid Blocks to Earthquakes. *Earthquake Engineering and Structural Dynamics*, 1980.

[18] M.J. DeJong and E.G. Dimitrakopoulos. Dynamically equivalent rocking structures. *Earthquake Engineering and Structural Dynamics*, 2014.

[19] E.G.Dimitrakopoulos and M.J. Dejong. Overturning of Retrofitted Rocking Structures under Pulse-Type Excitations. *Journal of Engineering Mechanics*, 2012.

[20] N. Makris J. Zhang. Rocking Response of Free-standing Blocks Under Cycloidal Pulses. *Journal of Engineering Mechanics*, 2001.

[21] N. Makris and Y. Roussos. Rocking response and overturning of equipment under horizontal pulse-type motions. *Rep. No. PEER- 98/05*, 1998.

[22] G.P. Mavroeidis and A.S. Papageorgiou. A Mathematical Representation of Near-Fault Ground Motions. *Bulletin of the Seismological Society in America*, 2003.